Catapults in SGD: spikes in the training loss and their impact on generalization through feature learning

Libin Zhu [1 2] **Chaoyue Liu** [2] **Adityanarayanan Radhakrishnan** [3 4] **Mikhail Belkin** [1 2]

Abstract

In this paper, we first present an explanation regarding the common occurrence of spikes in the training loss when neural networks are trained with stochastic gradient descent (SGD). We provide evidence that the spikes in the training loss of SGD are "catapults", an optimization phenomenon originally observed in GD with large learning rates in (Lewkowycz et al., 2020). We empirically show that these catapults occur in a low-dimensional subspace spanned by the top eigenvectors of the tangent kernel, for both GD and SGD. Second, we posit an explanation for how catapults lead to better generalization by demonstrating that catapults promote feature learning by increasing alignment with the Average Gradient Outer Product (AGOP) of the true predictor. Furthermore, we demonstrate that a smaller batch size in SGD induces a larger number of catapults, thereby improving AGOP alignment and test performance.

1. Introduction

Training algorithms are a key ingredient to the success of deep learning. Stochastic gradient descent (SGD) (Robbins & Monro, 1951), a stochastic variant of gradient descent (GD), has been effective in finding parameters that yield good test performance despite the complicated nonlinear nature of neural networks. Empirical evidence suggests that training networks using SGD with a larger learning rate results in better predictors (Frankle et al., 2020; Smith & Topin, 2019; Gilmer et al., 2021). In such settings, it is common to observe significant spikes in the training loss (LeCun et al., 2015; Ruder, 2016; Keskar & Socher, 2017; Xing et al., 2018) (see Fig. 1 as an example). One may not *a priori* expect the training loss to decrease back to its "pre-spike" level after a large spike. Yet, this is what is commonly observed in training. Furthermore, the resulting "post-spike" model can yield improved generalization performance (He et al., 2016; Zagoruyko & Komodakis, 2016; Huang et al., 2017).

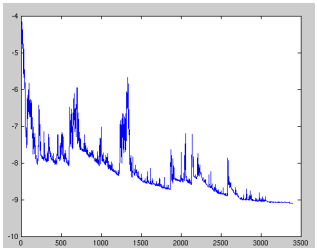


Figure 1. Spikes in training loss when optimized using SGD (x-axis: iteration). (Source: Wikipedia)

Why do spikes occur during training, and how do the spikes relate to generalization?

In this work, we answer these questions by connecting three common but seemingly unrelated phenomena in deep learning:

1. Spikes in the training loss of SGD,
2. Catapult dynamics in GD (Lewkowycz et al., 2020),
3. Better generalization when training networks with small batch SGD as opposed to larger batch size or GD.

In particular, we show that spikes in the training loss of SGD are caused by catapult dynamics, which were originally characterized in (Lewkowycz et al., 2020) as a single spike in the loss when training with GD and large learning rate. We then show that smaller batch size in SGD results in a greater number of catapults. We connect the optimization phenomena of catapults to generalization by showing that catapults improve generalization through increasing feature learning, which is quantified by the alignment between the Average Gradient Outer Product (AGOP) of the trained network and the true AGOP (Härdle & Stoker, 1989; Hristache et al., 2001; Xia et al., 2002; Trivedi et al., 2014; Radhakrishnan et al., 2024a). Since decreasing batch size in SGD leads to more catapults, our result implies that SGD with small batch size yields improved generalization (see Table 1

[1]Department of Computer Science, UCSD. [2]Halicioğlu Data Science Institute, UCSD. [3]Harvard University. [4]Broad Institute of MIT and Harvard. Correspondence to: Libin Zhu <libinzhu@ucsd.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

Batch size	AGOP alignment	Test loss
2000 (GD)	0.81	0.74
50	0.84	0.71
10	0.89	0.59
5	0.95	0.42

Table 1. Smaller SGD batch size leads to a higher (better) AGOP alignment and smaller (better) test loss. The results correspond to Fig. 9a (a synthetic dataset).

for an example). We outline our specific contributions in the context of optimization and generalization below.

Optimization. We demonstrate that spikes in the training loss, specifically measured by Mean Squared Error, occur in the top eigenspace of the Neural Tangent Kernel, a kernel resulting from the linearization of a neural network (Jacot et al., 2018). Namely, we project the residual (i.e., the difference between the predicted output and the target output) to the top eigenspace of the tangent kernel and show that spikes in the total loss function correspond to the spikes in the components of the loss in this low-dimensional subspace (see Section 3.1). In contrast, the components of the loss in the space spanned by the remaining eigendirections decrease monotonically. Thus, the catapult phenomenon occurs in the span of the top eigenvectors while the remaining eigendirections are not affected. This explains why the loss drops quickly to pre-spike levels, namely the loss value right before the spike, from the peak of the spike. We further show that multiple catapults can be generated in GD by increasing the learning rate during training (see Section 3.2). While prior work (Lewkowycz et al., 2020) observed that the spectral norm of the tangent kernel decreased for one catapult, we extend that observation by showing that the norm decreases after each catapult.

We further provide evidence for catapults in SGD with large learning rates (see Section 3.3). Namely, we demonstrate that spikes in the loss when training with SGD correspond to catapults by showing that similarly to GD:

1. The spikes occur in the top eigenspace of the tangent kernel,
2. Each spike results in a decrease in the spectral norm of the tangent kernel.

We corroborate our findings across several network architectures including Wide ResNet (Zagoruyko & Komodakis, 2016) and ViT (Dosovitskiy et al., 2021) and datasets including CIFAR-10 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011).

Moreover, as small batch size leads to higher variance in the eigenvalues of the tangent kernel for any given batch, small batch size results in an increased number of catapults.

Generalization. We posit that catapults improve the generalization performance by alignment between the AGOP of the trained network with that of the true model[1]. The AGOP identifies the features that lead to greatest change in predictor output when perturbed and has been recently posited as the mechanism through which neural networks learn features (Radhakrishnan et al., 2024a; Beaglehole et al., 2023). We use AGOP alignment to provide an explanation for prior empirical results from (Lewkowycz et al., 2020; Zhu et al., 2024) showing that a single catapult can lead to better test performance in GD. Moreover, we extend these prior results to show that test performance continues to improve as the number of catapults increases in GD. Thus, we show that decreasing batch size with SGD can lead to better test performance due to an increase in the number of catapults. We further demonstrate that AGOP alignment is an effective measure of generalization by showing that test error is highly correlated with the AGOP alignment when training on the same task across different optimization algorithms including Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012) and Adam (Kingma & Ba, 2015) etc. We corroborate our findings on CelebA (Liu et al., 2015) and SVHN (Netzer et al., 2011) datasets and architectures including fully-connected and convolutional neural networks. See Section 4.

1.1. Related works

Linear dynamics and catapult phase phenomenon. Recent studies have shown that (stochastic) GD for wide neural networks provably converges to global minima with an appropriately small learning rate (Du et al., 2019; Zou & Gu, 2019; Liu et al., 2022a). These works leveraged the fact that neural networks with sufficiently large widths, under specific initialization conditions, can be accurately approximated by their linearization obtained by the first-order Taylor expansion (Jacot et al., 2018; Liu et al., 2020; 2022b; Zhu et al., 2022). Therefore, their training dynamics are close to the dynamics of the corresponding linear models, under which the training loss decreases monotonically. Such a training regime is commonly referred to as the kernel regime. However, under the same setup of the kernel regime except using a large learning rate, GD will experience a catapult phase (Lewkowycz et al., 2020): the training loss increases drastically in the beginning stage of training then decreases, while GD still converges. Recent studies focusing on understanding catapults in GD include (Zhu et al., 2024), which considers quadratic approximations of neural networks, and (Meltzer & Liu, 2023), examining two-layer homogeneous neural networks. Our work investigates the impact of catapults in SGD on both optimization and generalization through experimental approaches.

[1]When the underlying model is not available, we use a SOTA model as a substitute.

Edge of stability. A phenomenon related to catapults is the "Edge of Stability" (EoS), which describes the dynamics of the training loss and the sharpness, i.e., eigenvalues of the Hessian of the loss, at the later stage of training networks with GD (Cohen et al., 2021) and SGD (Jastrzębski et al., 2019; Jastrzebski et al., 2020). There is a growing body of work analyzing the mechanism of EoS in training dynamics with GD (Arora et al., 2022; Ahn et al., 2022; Damian et al., 2023; Wang et al., 2022b; Agarwala et al., 2023; Agarwala & Dauphin, 2023; Wang et al., 2022a), and SGD (Kalra & Barkeshli, 2023). It was conjectured in (Cohen et al., 2021) that at EoS for GD the spikes in the training loss are micro-catapults. Our work provides evidence that the spikes in the training loss using SGD are catapults and demonstrates the connection between the loss spikes and feature learning.

Generalization and sharpness. It has been observed that networks trained with SGD generalize better than GD, and smaller batch sizes often lead to better generalization performance (LeCun et al., 2002; Keskar et al., 2017; Goyal et al., 2017; Jastrzębski et al., 2017; Masters & Luschi, 2018; Kandel & Castelli, 2020; Smith et al., 2021). Empirically, it has been observed that training with SGD results in flat minima (Hochreiter & Schmidhuber, 1994; 1997). However, we noticed that it is not always the case, e.g., (Geiping et al., 2022). A number of works been argued that flatness of the minima is connected to the generalization performance (Neyshabur et al., 2017; Wu et al., 2017; Kleinberg et al., 2018; Xie et al., 2021; Jiang* et al., 2020; Dinh et al., 2017), however we know only one theoretical result in that direction (Ding et al., 2024). Training algorithms aiming to find a flat minimum were developed and shown to perform well on a variety of tasks (Izmailov et al., 2018; Foret et al., 2021). As an explanation for empirically observed improved generalization, prior work (Lewkowycz et al., 2020) argued that a single catapult with GD resulted in flatter minima. In this work we propose a different line of investigation to understand generalization properties of GD-based algorithms based on feature learning as measured by the alignment with AGOP.

2. Preliminaries

Notation. We use bold letters (e.g., $\mathbf{w}$) to denote vectors and capital letters (e.g., K) to denote matrices. For a matrix, we use $\|\cdot\|_F$ to denote its Frobenius norm and use $\|\cdot\|_2$ to denote its spectral norm. For trainable parameters, we use superscript t, as in $\mathbf{w}^t$, to denote the time stamp during training. We use the big-O notation $O(\cdot)$ to hide constants, and use $\tilde{O}(\cdot)$ to further hide logarithmic factors. For a map $f(\mathbf{w}) : \mathbb{R}^p \to \mathbb{R}^c$, we use $\nabla_{\mathbf{w}} f(\mathbf{v})$ and $\nabla^2_{\mathbf{w}} f(\mathbf{v})$ to denote the first and second order derivative of f w.r.t. $\mathbf{w}$ evaluated at $\mathbf{v}$ respectively.

Optimization task. Consider a parameterized model $f(\mathbf{w}; \cdot) : \mathbb{R}^p \to \mathbb{R}$ (e.g., a neural network) with parameters $\mathbf{w}$ and a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ with data $\boldsymbol{x}_i \in \mathbb{R}^d$ and labels $y_i \in \mathbb{R}$ for $i \in [n]$. Denote $X \in \mathbb{R}^{n\times d}$ as the collection of training input data, with each row of X representing an input $\boldsymbol{x}_i$, and $\mathbf{y} := (y_1, \cdots, y_n)^T$. We further write $\mathbf{f} \in \mathbb{R}^n$ as the predictions of f on X. The goal of the optimization task is to minimize the Mean Square Error (MSE)

$$\mathcal{L}(\mathbf{w}; (X, \mathbf{y})) = \frac{1}{n}\sum_{i=1}^n (f(\mathbf{w}; \boldsymbol{x}_i) - y_i)^2 = \frac{1}{n}\|\mathbf{f} - \mathbf{y}\|^2. \quad (1)$$

Let $\mathbf{w}_0$ be the weight parameters at initialization. Mini-batch SGD is conducted as follows: at each step t, randomly sample a batch $\mathcal{B} \subset \mathcal{D}$ (of batch size b), and perform the update following

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \frac{\eta}{b}\frac{\partial}{\partial \mathbf{w}} \sum_{(\boldsymbol{x}_j, y_j)\in\mathcal{B}} (f(\mathbf{w}^t; \boldsymbol{x}_j) - y_j)^2,$$

where η is the learning rate. When $b = n$, mini-batch SGD reduces to the full-batch gradient descent (GD).

Neural Tangent Kernel (NTK). Proposed in (Jacot et al., 2018), NTK is a useful tool in understanding and analyzing over-parameterized neural networks.

Definition 1 ((Neural) Tangent Kernel)**.** *The (neural) tangent kernel $K(\mathbf{w}; \cdot, \cdot)$ for a parameterized machine learning model $f(\mathbf{w}; \cdot) : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}$ is defined as:*

$$\forall \boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^d, \quad K(\mathbf{w}; \boldsymbol{x}, \boldsymbol{z}) = \left\langle \frac{\partial f(\mathbf{w}; \boldsymbol{x})}{\partial \mathbf{w}}, \frac{\partial f(\mathbf{w}; \boldsymbol{z})}{\partial \mathbf{w}} \right\rangle.$$

Given the training data inputs X, the NTK can be evaluated on any pair of inputs $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which results in a $n \times n$ matrix K, called the NTK matrix. By definition, the NTK matrix K is symmetric and positive semi-definite. Therefore, it can be decomposed as $K = \sum_{j=1}^n \lambda_j \mathbf{u}_j \mathbf{u}_j^T$, with $\lambda_j \in \mathbb{R}$ and $\mathbf{u}_j \in \mathbb{R}^n$, $j \in \{1, \cdots, n\}$, being the eigenvalues and unit-length eigenvectors, respectively. Without loss of generality, we assume $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$.

Top-eigenspace and decomposition of loss function. Given an integer s, $1 \leq s < n$, we call the *top eigenspace* (or *top-s eigenspace*) of NTK as the subspace spanned by the top eigenvectors $\mathbf{u}_j$ with $1 \leq j \leq s$. We also define projection operators $\mathcal{P}_{\leq s} : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathcal{P}_{>s} : \mathbb{R}^n \to \mathbb{R}^n$, such that for any vector $\mathbf{v} \in \mathbb{R}^n$ the followings hold:

$$\mathcal{P}_{\leq s}\mathbf{v} = \sum_{i=1}^s \langle \mathbf{v}, \mathbf{u}_i \rangle \mathbf{u}_i, \quad \mathcal{P}_{>s}\mathbf{v} = \sum_{i=s+1}^n \langle \mathbf{v}, \mathbf{u}_i \rangle \mathbf{u}_i.$$

The MSE Eq. (1) can be decomposed as

$$\mathcal{L} = \frac{1}{n}\|\mathbf{f} - \mathbf{y}\|_2^2 = \frac{1}{n}\|\mathcal{P}_{\leq s}(\mathbf{f} - \mathbf{y})\|_2^2 + \frac{1}{n}\|\mathcal{P}_{>s}(\mathbf{f} - \mathbf{y})\|_2^2 =: \mathcal{L}_{\leq s} + \mathcal{L}_{>s}. \quad (2)$$

Critical learning rate. When a constant learning rate of the algorithm is used throughout the training, it is important to select the learning rate η, as a large η easily leads to a divergence of loss and a small η slows down the training procedure. A conventional wisdom is to set η no larger than the *critical learning rate* $\eta_{\text{crit}}(\mathbf{w}) := \frac{2}{\lambda_{\max}(H_{\mathcal{L}}(\mathbf{w}))}$, where $H_{\mathcal{L}} := \nabla^2_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ denotes the Hessian of the loss. This intuition follows from the well-known lemma in optimization:

Lemma 1 (Descent Lemma (Nesterov, 1983))**.** *For a smooth loss $\mathcal{L}(\mathbf{w}) : \mathbb{R}^p \to \mathbb{R}$, suppose $\lambda_{\max}(H_{\mathcal{L}}(\mathbf{w})) \leq \beta$ for all $\mathbf{w} \in \mathbb{R}^p$, then GD satisfies:*

$$\mathcal{L}(\mathbf{w}^{t+1}) \leq \mathcal{L}(\mathbf{w}^t) - \eta(1 - \eta\beta/2)\left\|\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}^t)\right\|^2.$$

For $\eta < 2/\beta$, the descent lemma guarantees the decrease of the loss. Note that this inequality is tight for quadratic loss, e.g., loss for linear models. For neural networks with sufficient width, due to *transition to linearity (Liu et al., 2020)*, the critical learning rate η_{crit} almost does not change during training, and η_{crit} can be well-approximated by NTK (exact, for linear models): $\eta_{\text{crit}} \approx n/\|K\|_2 = n/\lambda_1$ (see the derivation in Appendix A.1). For neural networks that are not wide, (Papyan, 2019; Agarwala et al., 2023; Wang et al., 2022b) showed the approximation still holds and we provide additional evidence for SGD in Appendix A.2.

Note that unless specified, the critical learning rate is evaluated at initialization $\mathbf{w}_0$.

Catapult dynamics. It was recently observed in (Lewkowycz et al., 2020) that, for wide neural network, full batch GD with a learning rate that is larger than η_{crit} (e.g., $\eta \in (\eta_{\text{crit}}, 2\eta_{\text{crit}})$ as shown in (Lewkowycz et al., 2020)) surprisingly ends up with a convergence. Instead of the expected divergence, the loss decreases after a drastic increase at the beginning stage of training, forming a loss spike (see Fig. 2). Moreover, $\|K\|_2$ is observed to be smaller at the end of the spike. Interestingly, the solution found by this large-learning-rate GD turns out to perform better in terms of test loss. Intuitively, the decrease in $\|K\|_2$ raises the divergence threshold $n/\|K\|_2$ which allows a final convergence.

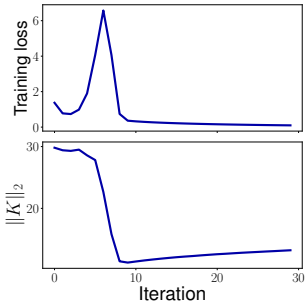
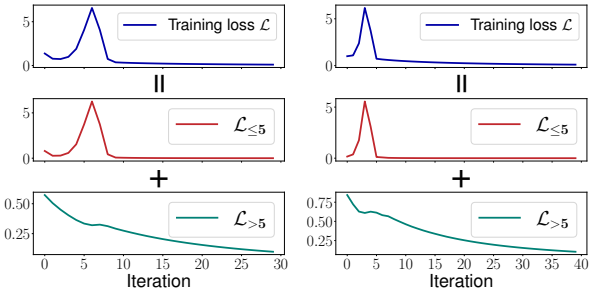


Figure 2. An illustration of the catapult. This experiment corresponds to Fig. 3a.

In this paper, we refer *catapult dynamics* as the phenomenon of a drastic increase followed by a fast decrease in the training loss which is triggered by a learning rate larger than η_{crit} and accompanied by a decreasing $\|K\|_2$.



(a) Loss decomposition(FCN) (b) Loss decomposition(CNN)

Figure 3. **Catapult occurring in the top eigenspace of NTK in GD for 5-layer FCN (a) and CNN (b).** The training loss is decomposed into the eigenspace of NTK, i.e., $\mathcal{L} = \mathcal{L}_{\leq 5} + \mathcal{L}_{>5}$. In the experiment, both networks are trained by GD on 128 data points from CIFAR-10 with learning rate 6 and 8 respectively (the critical learning rates are 3.6 for FCN and 4.5 for CNN).

3. Catapults in optimization

3.1. Catapults occur in the top eigenspace of the tangent kernel for GD

The training dynamics of the machine learning model, e.g., a neural network, are closely related to its NTK $K^t := K(\mathbf{w}^t; X, X) \in \mathbb{R}^{n\times n}$. Specifically, when the loss is optimized by gradient flow (continuous-time GD) with learning rate η, the output follows the dynamic equation (Lee et al., 2019):

$$d\mathbf{f}^t/dt = -2\eta K^t/n(\mathbf{f}^t - \mathbf{y}).$$

By discrete time GD, this becomes

$$\mathbf{f}^{t+1} - \mathbf{y} = \left(I_n - 2\eta K^t/n\right)(\mathbf{f}^t - \mathbf{y}) + \Delta_{H^t_{\mathbf{f}}}, \quad (3)$$

with $\Delta_{H^t_{\mathbf{f}}} := \left\langle \mathbf{w}^{t+1} - \mathbf{w}^t, \nabla^2_{\mathbf{w}}\mathbf{f}(\xi)(\mathbf{w}^{t+1} - \mathbf{w}^t)\right\rangle \in \mathbb{R}^n$ and $\xi = \tau\mathbf{w}^t + (1-\tau)\mathbf{w}^{t+1}, \tau \in (0,1)$. Note that for finitely wide neural networks, $\left\|\Delta_{H^t_{\mathbf{f}}}\right\|_2$ is small compared to the first term (Papyan, 2019; Wang et al., 2022b) and is exactly zero for infinitely wide neural networks (Lee et al., 2019). Therefore, the training dynamics of neural networks are mainly determined by the first term in R.H.S. of the above equation, which relies on the spectral information of the NTK K^t. This data-dependent NTK is also useful for understanding the generalization performance of neural networks (Fort et al., 2020; Atanasov et al., 2022; Ortiz-Jiménez et al., 2021; Loo et al., 2022).

Consider decomposing Eq. (3) into eigendirections of the NTK K^t, i.e., $\langle \mathbf{f}^t - \mathbf{y}, \mathbf{u}^t_i\rangle$. Supposing the dynamics among eigendirections are not interacting and $\mathbf{u}_i$ is constant, we expect that the increase of training loss during catapult occurs in the top few eigendirections where $\eta > n/\lambda_i$, while the loss corresponding to the remaining eigendirections remain

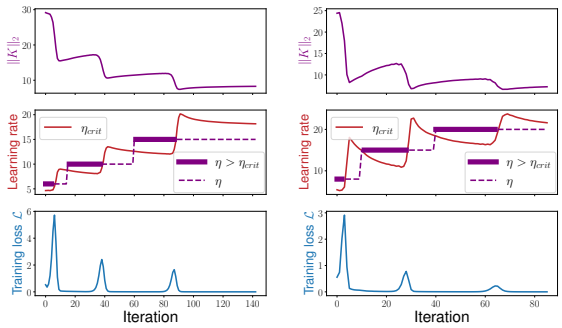
decreasing. Indeed, this has been theoretically shown to be true on quadratic models that approximate wide neural networks (Zhu et al., 2024).

Claim 1. *The catapult occurs in the top eigenspace of the tangent kernel: the loss component corresponding to the top-s eigenspace $\mathcal{L}_{\leq s}$ has a spike during the catapult, while the loss component in the complementary eigenspace $\mathcal{L}_{>s}$ decreases monotonically.*

Remark 1. *We note that catapults does not occur in all eigendirections, as the learning rate η cannot be arbitrarily large. Instead, there is a maximum learning rate $\eta_{\max}$ such that if $\eta > \eta_{\max}$ the algorithm will diverge. For instance, $\eta_{\max} = 2\eta_{\text{crit}}$ for quadratic models (Zhu et al., 2024) and $\eta_{\max} \approx 6\eta_{\text{crit}}$ for ReLU networks (Lewkowycz et al., 2020). Therefore, for any learning rate $\eta \in (\eta_{\text{crit}}, \eta_{\max})$ such that catapult occurs, only the top few eigendirections satisfy $n/\lambda_i < \eta_{\max}$. We consistently observe that s is a small constant no larger than 10 in all our experiments.*

We empirically justify Claim 1 for neural networks. In particular, we consider three neural network architectures: a 5-layer Fully Connected Neural Network (FCN), a 5-layer Convolutional Neural Network (CNN), and Wide ResNets 10-10; and three datasets CIFAR-10, SVHN, and a synthetic dataset. The details of experimental setup can be found in Appendix F. We present a selection of the results in Fig. 3 with the remaining results in Fig. 14 and 15 in Appendix B. We can see that $\mathcal{L}_{\leq 5}$ corresponds to the spike in the training loss while $\mathcal{L}_{>5}$ decreases almost monotonically. Concurrently with this study, (Zhang & Xu, 2023) showed that the loss spike in GD is primarily due to the low-frequency component, corroborating our findings through a frequency perspective.

We note that the same phenomenon holds for multidimensional outputs. See more details in Fig. 16 in Appendix B.



(a) 5-layer FCN (b) 5-layer CNN

Figure 4. **Multiple catapults during GD with increased learning rates**. We train a 5-layer FCN and CNN on a subset of CIFAR-10 using GD. The learning rate is increased two times for each experiment. The experimental details can be found in Appendix F.2.

3.2. Inducing multiple catapults in GD

While prior work showed a single catapult during training with gradient descent (Lewkowycz et al., 2020; Zhu et al., 2024; Kalra & Barkeshli, 2023), we present that catapults can be induced multiple times by repeatedly increasing the learning rate during training.

Specifically, during a catapult, the norm of NTK $\|K\|_2$ decreases, which leads to an increase in the critical learning rate $\eta_{\text{crit}} \approx n/\|K\|_2$, see Fig. 4. When the loss starts to decrease during a catapult, η_{crit} surpasses the current learning rate η of the algorithm. Hence, after each catapult, one can reset the algorithmic learning rate η to be greater than the current η_{crit} to trigger another catapult. In practice, we observe that a sequence of catapults can be triggered by repeating the above procedure. See Fig. 4 for a demonstration of various neural network architectures.

Interestingly, with multiple catapults, the gradient descent can ultimately converge with a much larger learning rate, which leads to a divergence, instead of a catapult, if set as the initial learning rate of gradient descent (see Fig. 18 in Appendix B.3). Furthermore, thanks to the relation $\eta_{\text{crit}} \approx n/\|K\|_2$, this indicates that the multiple catapults achieve a much smaller $\|K\|_2$ which can not be obtained in the scenario of a single catapult. See Fig. 4 for an experimental demonstration. Moreover, the multiple catapults lead to better generalization performance than a single catapult. We defer this discussion of generalization performance to Section 4.

3.3. Catapults in SGD

In this section, we consider the stochastic setting, and argue that the spikes often observed in the training loss of SGD (e.g., Fig. 1) are in fact *catapults*.

Mechanism of catapults in SGD. Recall that the catapults are triggered when $\eta > \eta_{\text{crit}}$. Unlike in deterministic gradient descent, the mini-batch stochastic training dynamics is determined by the NTK matrix evaluated on the given batch X_{batch}. Specifically, the update equation of mini-batch SGD becomes (c.f. Eq.(3) of GD):

$$\mathbf{f}^{t+1}_{\text{batch}} - \mathbf{y}_{\text{batch}} = (I_b - \eta K_t(X_{\text{batch}})/b)\,(\mathbf{f}^{t}_{\text{batch}} - \mathbf{y}_{\text{batch}}) + \Delta_{H_{\mathbf{f}^t_{\text{batch}}}}, \quad (4)$$

where b is the mini batch size, $\mathbf{f}_{\text{batch}} := \mathbf{f}(X_{\text{batch}})$ and $\mathbf{y}_{\text{batch}}$ is the label corresponds to X_{batch}. It is important to note that in mini-batch SGD the critical learning rate $\eta_{\text{crit}}(X_{\text{batch}})$ becomes batch dependent: for batches that have relatively large (small, respectively) $\|K_{\text{batch}}\|_2$, the corresponding critical learning rate $\eta_{\text{crit}}(X_{\text{batch}})$ is relatively small (large, respectively). Then, if $\eta_{\text{crit}}(X_{\text{batch}})$ of a given batch is smaller than the algorithmic learning rate

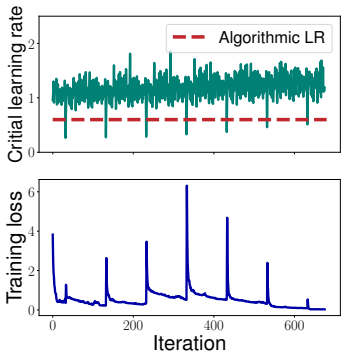


Figure 5. **Exact match between the occasion when $\eta > \eta_{\text{crit}}(X_{\text{batch}})$ and loss spike for SGD.** We train a two-layer neural network on a synthetic dataset using SGD with batch size one.

η of SGD, we expect a catapult will happen: an increase in the running training loss.

Indeed, this expectation is confirmed in our experiments. Specifically, we train the network on a synthetic dataset with SGD and consider batch size one. We set the algorithmic learning rate higher than the critical learning rate for only one training example. As expected, we observe that the loss spikes only occur when the gradient is computed based on that particular training example. See the results in Fig. 5 and the detailed experimental setup in Appendix F.3.

In more practical scenarios, we train a shallow network by SGD with mini-batch size 32, on a subset of CIFAR-10 with training size 128. First, when the algorithmic learning rate η is smaller than $\eta_{\text{crit}}(X_{\text{batch}})$ of all the batches (as shown in the case of $\eta = 0.1$ in (Fig. 6 upper left)), we observe that the training loss of mini-batch SGD monotonically decreases until convergence without any spike; when η becomes greater than $\eta_{\text{crit}}(X_{\text{batch}})$ for some of the batches (as shown in the case of $\eta = 0.8$ in Fig. 6 upper right), many spikes appear in the training loss. Moreover, we show

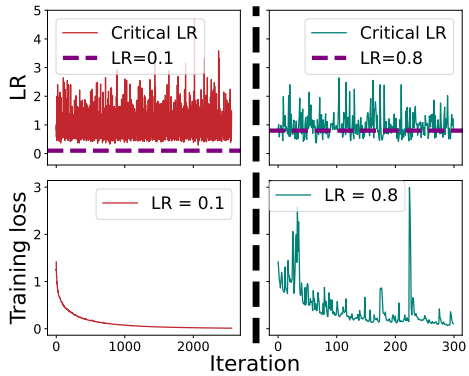


Figure 6. **Mechanism of catapults in SGD.** LR denotes learning rate. We train a two-layer neural network on 128 data points of CIFAR-10 using SGD with batch size 32. The leaning rate is 0.1 for the left two panels and 0.8 for the right two panels.

Network Architecture	Match rate between $\Delta\mathcal{L} > 0$ and $\eta > \eta_{\text{crit}}(X_{\text{batch}})(\%)$
Shallow network	97.32 ± 0.45
5-layer FCN	96.17 ± 1.46
5-layer CNN	94.67 ± 3.27

Table 2. **The match rate between $\Delta\mathcal{L}^t := \mathcal{L}^{t+1} - \mathcal{L}^t > 0$ and $\eta > \eta^t_{\text{crit}}(X_{\text{batch}})$.** For each network architecture, we calculate the match rate as the ratio of occurrences where $\eta > \eta^t_{\text{crit}}(X_{\text{batch}})$ for all t such that $\mathcal{L}^{t+1} > \mathcal{L}^t$ until convergence of SGD (see the training loss in Fig. 6(c) for shallow net and Fig. 7 (a,b) for deep nets). Each result is the average of 3 independent runs.

that these spikes in the (total) training loss are caused by large learning rates for batches. Specifically, for the case of $\eta = 0.8$, we verify that whenever the (total) training loss increases, the algorithmic learning rate η is larger than the critical learning rate $\eta_{\text{crit}}(X_{\text{batch}})$ for the current batch X_{batch}. This phenomenon is further verified for 5-layer FCN and CNN. See Table 2.

Decreases in the spectral norm of the tangent kernel correspond to spikes. As shown in prior work (Lewkowycz et al., 2020) and in the multiple catapults in Section 3.2 an important characterization of the catapult dynamics is the decreasing NTK norm $\|K\|_2$. Here, we experimentally show that the spectral norm of the NTK decreases whenever there is a spike in the SGD training loss.

Specifically, we consider four network architectures: (1) 5-layer FCN, (2) 5-layer CNN (the same as the ones in Fig. 3), (3) Wide ResNets 10-10 and (4) ViT-4. We train neural networks on a subset of CIFAR-10 using SGD. Fig. 7 shows some of the results (more results on various datasets and parameterizations are available in Appendix C). One can easily see that at each spike of the training loss, there is a significant drop in the NTK spectral norm $\|K\|_2$, while $\|K\|_2$ are mostly increasing or staying unchanged at other steps. This empirical evidence corroborates that these spikes are indeed (mini-)catapults, instead of some random fluctuations in the training loss. All experimental details can be found in Appendix F.

Catapults occur in the top eigenspace of the tangent kernel for SGD. As discussed in Section 3.1, another characteristic of the catapults is that they occur in the top eigenspace of the tangent kernel. We show that these loss spikes in SGD also occur in the top eigenspace, as another evidence that these spikes are catapults.

In the experiments, we decompose the training loss of SGD into $\mathcal{L}_{\leq 1}$ and $\mathcal{L}_{>1}$ based on the eigendirections of the tangent kernel. We observe that $\mathcal{L}_{\leq 1}$ corresponds to the spikes in the training loss, while the decrease of $\mathcal{L}_{>1}$ is nearly

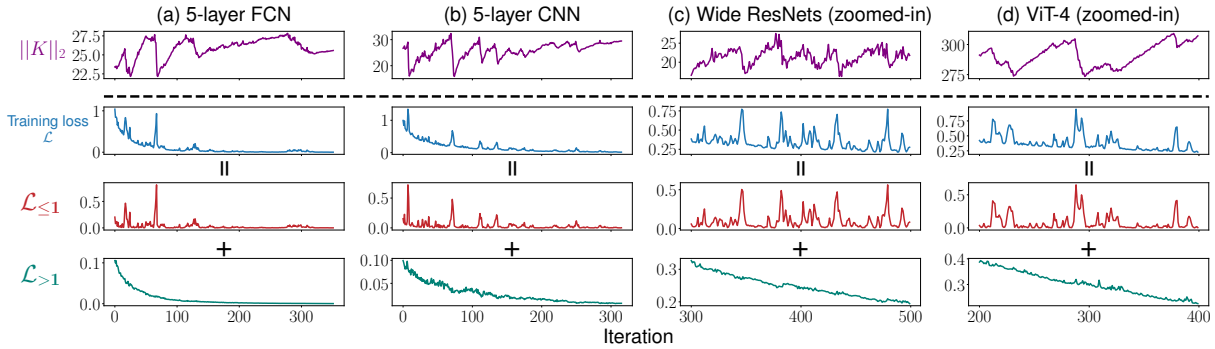


Figure 7. **Catapult dynamics in SGD for modern deep architectures.** The training loss is decomposed based on the eigendirections of the NTK: $\mathcal{L}_{\leq 1}$ and $\mathcal{L}_{>1}$. We train the networks on a subset of CIFAR-10 using SGD. The complete versions of Panel (c) and (d) can be found in Fig. 20 in Appendix C.

monotonic, with only small oscillations present. See Fig. 7 for deep networks and Fig. 19 in Appendix C for the shallow network with $\eta = 0.8$ corresponding to Fig. 6.

This observation, along with the results that the spectral norm of the NTK decreases corresponding to the loss spike, is consistent with our findings for GD and provides evidence that the spikes in training loss for neural networks are caused by catapults.

Remark 2 (Top eigenspace accounts for the sharp loss spikes in SGD)**.** *In SGD training loss, the sharp spikes we observe last only a few iterations before rapidly returning to their pre-spike levels. These spikes can be attributed to catapults occurring in the top-1 eigendirection of the tangent kernel. Consider the loss change in each eigendirection of the tangent kernel. We expect that the rate of loss change in each eigendirection depends on the corresponding eigenvalue's size. Therefore, with a constant learning rate, changes happen faster in the top eigendirections, which accounts for the sharp loss spikes in SGD as they occur in the top-1 eigendirection.*

Remark 3 (Catapults in SGD with cyclical learning rate schedule)**.** *Training neural networks with the learning rate cyclically varying between selected boundary values was widely shown to improve the generalization performance of neural networks with less tuning (Izmailov et al., 2018; Smith, 2017). We empirically show that the increasing phase of the cyclical learning rate schedule induces catapults in SGD. Specifically, we observe that there is a spike in the training loss when the learning rate is increased. We demonstrate that the loss spikes are caused by catapults, by providing similar evidence to the case of SGD with a constant learning rate. See the results in Fig. 24 in Appendix C.5.*

4. Catapults lead to better generalization through feature learning

Previous empirical results from (Lewkowycz et al., 2020; Zhu et al., 2024) show that a single catapult can lead to better test performance in GD for wide neural networks. In this section, we observe a similar trend in our experiments for both GD and SGD with multiple catapults. We posit an explanation for this phenomenon by demonstrating that catapults improve feature learning by increasing alignment between the Average Gradient Outer Products (AGOP) of the trained network and the true model, therefore improving generalization. We formalize this claim as follows. Let $\{(x_i, f^*(x_i))\}_{i=1}^n$ denote training data with $f^*(x)$ denoting the true model. Then, for any predictor f, the AGOP, $G(f, \{x_1, \ldots, x_n\})$ is given as follows:

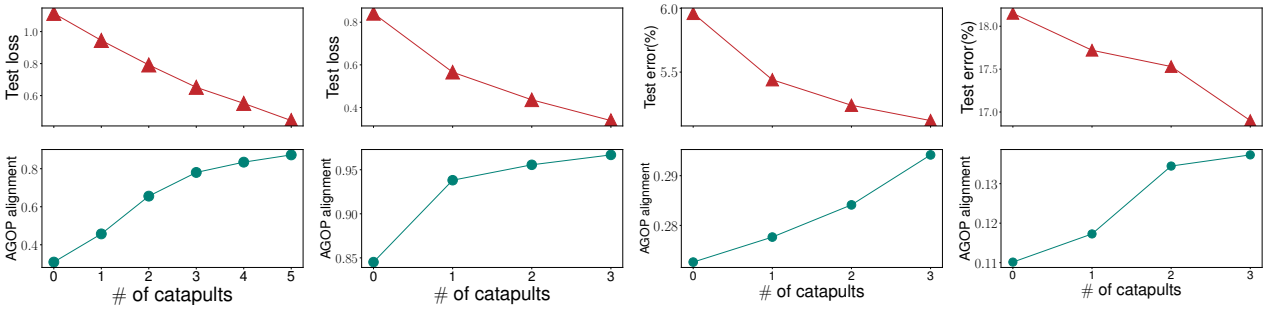
$$G(f, \{x_1, \ldots, x_n\}) = \frac{1}{n} \sum_{i=1}^{n} \nabla_x f(x_i) \nabla_x f(x_i)^T \; ; \quad (5)$$

where $\nabla_x f$ denotes the gradient of f with respect to the input x.[2] We will suppress the dependence on the data $\{x_i\}_{i=1}^n$ to simplify notation. Assuming the data x_i are i.i.d. samples from an underlying data distribution, in the limit as $n \to \infty$, Eq. (5) converges to a quantity referred to as the Expected Gradient Outer Product (EGOP). Letting G^* denote the EGOP of f^* and G denote the AGOP of f, we define *AGOP alignment* using the cosine similarity between G, G^* as follows:

$$\textbf{AGOP alignment} : \cos(G, G^*) := \frac{\mathrm{Tr}\left(G^T G^*\right)}{\|G\|_F \|G^*\|_F}. \quad (6)$$

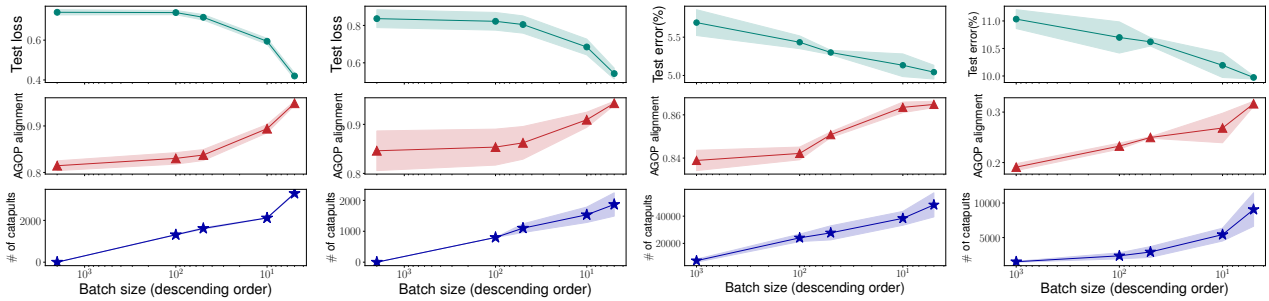
Remark 4. *G^* captures the directions along which f^* varies the most and those along which it varies least. When training a predictor on data generated using low rank G^*, it is possible improve sample efficiency by first estimating G^*. Indeed, this has been theoretically shown in the case of multi-index models, i.e., functions of the form $f^*(\boldsymbol{x}) = g(U\boldsymbol{x})$ where the index space U is a low-rank matrix (Härdle & Stoker, 1989; Trivedi et al., 2014; Yuan et al., 2023). Additionally, a recent line of work connected*

[2]For predictors with multivariate outputs, we consider the Jacobian instead of the gradient.



(a) Rank-2 regression (b) Rank-3 regression (c) SVHN (d) CelebA

Figure 8. **Multiple catapults in GD.** A greater number of catapults in GD leads to a higher (better) AGOP alignment and smaller (better) test loss/error. We train 2-layer FCN in Panel(a), 4-layer FCN in Panel(b,d) and 5-layer CNN in Panel(c). Experimental details can be found in Appendix F.4.



(a) Rank-2 regression (b) Rank-3 regression (c) SVHN (d) CelebA

Figure 9. **Similarity between AGOP alignment and generalization.** A greater number of catapults in SGD leads to a higher (better) AGOP alignment and smaller (better) test loss/error. We train a 2-layer FCN in Panel(a), 4-layer FCN in Panel(b,d) and 5-layer CNN in Panel(c) by SGD.

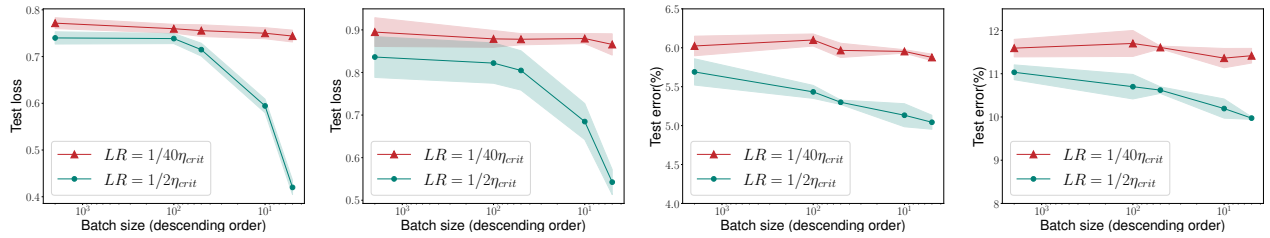
AGOP with feature learning in neural networks and further demonstrated that training predictors on data transformed by AGOP can lead to substantial increases in test performance (Radhakrishnan et al., 2024a; Beaglehole et al., 2023; Radhakrishnan et al., 2024b). Thus, we believe that AGOP alignment is a key measure for generalization, and we next corroborate our claim empirically across a broad class of network architectures and prediction tasks.

Experimental settings. We work with a total of seven datasets: three synthetic datasets and four real-world datasets. For synthetic datasets, we consider true functions $f^*(\boldsymbol{x}) = (1)x_1x_2$ (rank-2), $(2)x_1x_2(\sum_{i=1}^{10} x_i)$ (rank-3) and $(3)\sum_{j=1}^{4}\prod_{i=1}^{j} x_i$ (rank-4) (Abbe et al., 2021). For the four real-world datasets, we use (1) CelebA (Liu et al., 2015), (2) SVHN dataset (Netzer et al., 2011), (3) Fashion-MNIST (Xiao et al., 2017) and (4) USPS dataset (Hull, 1994). When the underlying model is not available, we use a state-of-the-art model as a substitute. We present the results for a selection of the datasets in this section and put the results for the remaining datasets in Appendix E.3. The experimental details can be found in Appendix F.

Improved test performance by catapults in GD. In Section 3.2, we showed that catapults can be generated multiple times. We now show that generating multiple catapults leads to improved test performance of neural networks trained with GD by leading to increased AGOP alignment. In Fig. 8, we can see for all tasks, the test loss/error decreases as the number of catapults increases while AGOP alignment increases. This indicates that learning the EGOP strongly correlates with test performance.

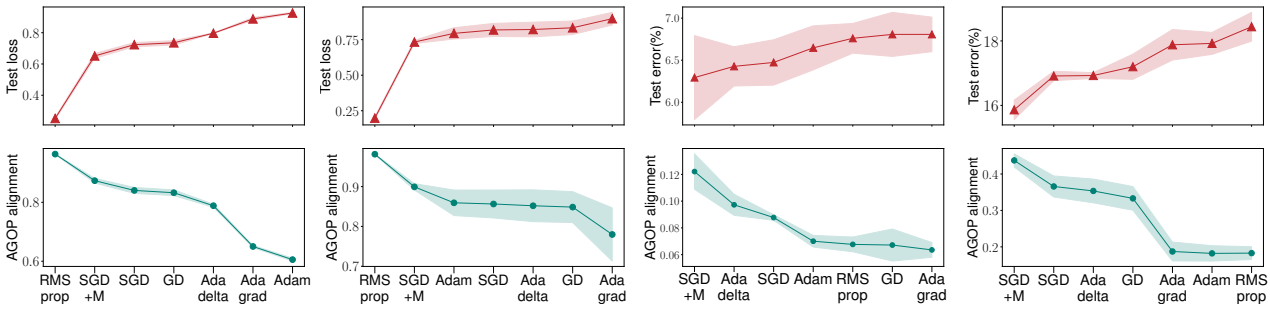
Remark 5. *As discussed earlier, AGOP alignment is a means of improving sample efficiency when training on data from multi-index models with low-rank index space. Our results on synthetic datasets show that catapults increase AGOP alignment, thereby leading to improved test performance. Additionally, we show that when the index space is full-rank, which can be effectively learned by neural networks in the NTK regime, catapults do not improve the test performance as well as the AGOP alignment. See Fig. 29 in Appendix D.*

Improved test performance by catapults in SGD. In Section 3.3, we have demonstrated the occurrence of catapults in SGD. We now show that decreasing batch size



(a) Rank-2 regression (b) Rank-3 regression (c) SVHN (d) CelebA

Figure 10. **"Large" vs. "small" learning rate on test performance with different batch sizes.** We consider the same setting as in Fig. 9 except for selecting a smaller learning rate $\eta_{\text{crit}}/40$ compared to $\eta_{\text{crit}}/2$ in Fig. 9. Here η_{crit} is the critical learning rate for the whole dataset.



(a) Rank-2 regression (b) Rank-3 regression (c) SVHN (d) CelebA

Figure 11. **Strong correlation between generalization and AGOP alignment for different optimization algorithms.** We train a 2-layer FCN in Panel(a), 4-layer FCN in Panel(b,d) and 5-layer CNN in Panel(c). We use GD, SGD, SGD with Momentum (Qian, 1999)(SGD+M), RMSprop (Hinton, 2014), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012) and Adam (Kingma & Ba, 2015) for training.

in SGD leads to better test performance as a result of an increase in the number of catapults and thus, increased AGOP alignment. We estimate the number of catapults during training by counting the number of the occurrence of the event $\eta - \eta_{\text{crit}}(X_{\text{batch}}) > \epsilon$ with $\epsilon = 10^{-8}$ until the best validation loss/error.

In Fig. 9, we can see that across all tasks, as the batch size decreases, (1) the number of catapults increases, (2) the test loss/error decreases and (3) the AGOP alignment improves. These findings indicate that in SGD, a smaller batch size leads to more catapults which in turn improves the test performance through alignment with the AGOP. These observations are consistent with our findings in GD.

Batch size does not affect generalization when the learning rate is small. Given the discussion above, sufficiently small learning rates will result in no catapults for any batch size. Thus we expect that all batch sizes will provide similar generalization performance for sufficiently small learning rates. This, indeed, is what we observe in the experiments presented in Fig. 10 where we keep the same experimental setting as for Fig. 9 except for a smaller learning rate. Specifically, we observe that while decreasing batch size consistently improves generalization for large learning rates, it has little effect on generalization for small learning rates.

Generalization with different optimizers correlates with AGOP alignment. We further demonstrate the strong correlation between the test performance and AGOP alignment by comparing the predictors trained on the same task with different optimization algorithms. From the results shown in Fig. 11, we can see that the AGOP alignment strongly correlates with the test performance, which suggests that models learning the AGOP is useful for learning the problem.

5. Conclusions

In this work, we addressed the following questions: (1) why do spikes in training loss occur during training with SGD and (2) how do the spikes relate to generalization? For the first question, we demonstrate that the spikes in the training loss are caused by the catapult dynamics in the top eigenspace of the tangent kernel. For the second question, we show that catapults lead to increased alignment between the AGOP of the model being trained and the AGOP of the underlying model or its state-of-the-art approximation. A consequence of our results is the explanation for the observation that SGD with small batch size often leads to improved generalization. This is due to an increase in the number of catapults for small batch sizes, due to increased batch variability, which, in turn, leads to better AGOP alignment.

Impact Statement

Our work sheds light on certain phenomena which have been empirically in neural networks. We hope it will be useful for developing new methods and analyses. The authors are unable to foresee the potential societal consequences of this particular work within the broad landscape of future AI.

Acknowledgements

A.R. is supported by the Eric and Wendy Schmidt Center at the Broad Institute. We are grateful for the support from the National Science Foundation (NSF) and the Simons Foundation for the Collaboration on the Theoretical Foundations of Deep Learning (https://deepfoundations.ai/) through awards DMS-2031883 and #814639 and the TILOS institute (NSF CCF-2112665). This work used NVIDIA V100 GPUs NVLINK and HDR IB (Expanse GPU) at SDSC Dell Cluster through allocation TG-CIS220009 and also, Delta system at the National Center for Supercomputing Applications through allocation bbjr-delta-gpu from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

Abbe, E., Boix-Adsera, E., Brennan, M. S., Bresler, G., and Nagaraj, D. The staircase property: How hierarchical structure can guide deep learning. *Advances in Neural Information Processing Systems*, 34:26989–27002, 2021.

Agarwala, A. and Dauphin, Y. Sam operates far from home: eigenvalue regularization as a dynamical phenomenon. In *International Conference on Machine Learning*, pp. 152–168. PMLR, 2023.

Agarwala, A., Pedregosa, F., and Pennington, J. Second-order regression models exhibit progressive sharpening to the edge of stability. In *International Conference on Machine Learning*, pp. 169–195. PMLR, 2023.

Ahn, K., Zhang, J., and Sra, S. Understanding the unstable convergence of gradient descent. In *International Conference on Machine Learning*, pp. 247–257. PMLR, 2022.

Arora, S., Li, Z., and Panigrahi, A. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pp. 948–1024. PMLR, 2022.

Atanasov, A., Bordelon, B., and Pehlevan, C. Neural networks as kernel learners: The silent alignment effect. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=1NvflqAdoom.

Banerjee, A., Cisneros-Velarde, P., Zhu, L., and Belkin, M. Neural tangent kernel at initialization: linear width suffices. In *Uncertainty in Artificial Intelligence*, pp. 110–118. PMLR, 2023.

Beaglehole, D., Radhakrishnan, A., Pandit, P., and Belkin, M. Mechanism of feature learning in convolutional neural networks. *arXiv preprint arXiv:2309.00570*, 2023.

Cohen, J., Kaur, S., Li, Y., Kolter, J. Z., and Talwalkar, A. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Damian, A., Nichani, E., and Lee, J. D. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nhKHA59gXz.

Ding, L., Drusvyatskiy, D., Fazel, M., and Harchaoui, Z. Flat minima generalize for low-rank matrix recovery. *Information and Inference: A Journal of the IMA*, 13(2): iaae009, 2024.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Du, S., Lee, J., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pp. 1675–1685. PMLR, 2019.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

Fort, S., Dziugaite, G. K., Paul, M., Kharaghani, S., Roy, D. M., and Ganguli, S. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33: 5850–5861, 2020.

Frankle, J., Schwab, D. J., and Morcos, A. S. The early phase of neural network training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkl1iRNFwS.

Geiping, J., Goldblum, M., Pope, P., Moeller, M., and Goldstein, T. Stochastic training is not necessary for generalization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=ZBESeIUB5k.

Gilmer, J., Ghorbani, B., Garg, A., Kudugunta, S., Neyshabur, B., Cardoze, D., Dahl, G., Nado, Z., and Firat, O. A loss curvature perspective on training instability in deep learning. *arXiv preprint arXiv:2110.04369*, 2021.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Härdle, W. and Stoker, T. M. Investigating smooth multiple regression by the method of average derivatives. *Journal of the American statistical Association*, 84(408):986–995, 1989.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., 2014. URL http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Hochreiter, S. and Schmidhuber, J. Simplifying neural nets by discovering flat minima. *Advances in neural information processing systems*, 7, 1994.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural computation*, 9(1):1–42, 1997.

Hristache, M., Juditsky, A., Polzehl, J., and Spokoiny, V. Structure adaptive approach for dimension reduction. *Annals of Statistics*, pp. 1537–1566, 2001.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Hull, J. J. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. doi: 10.1109/34.291440.

Izmailov, P., Wilson, A., Podoprikhin, D., Vetrov, D., and Garipov, T. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 876–885, 2018.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho*, K., and Geras*, K. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1g87C4KwB.

Jastrzębski, S., Kenton, Z., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. On the relation between the sharpest directions of DNN loss and the SGD step length. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SkgEaj05t7.

Jiang*, Y., Neyshabur*, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

Kalra, D. S. and Barkeshli, M. Phase diagram of early training dynamics in deep neural networks: effect of the learning rate, depth, and width. *Advances in Neural Information Processing Systems*, 37, 2023.

Kandel, I. and Castelli, M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT express*, 6(4):312–315, 2020.

Keskar, N. S. and Socher, R. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1oyRlYgg.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

Kleinberg, B., Li, Y., and Yuan, Y. An alternative view: When does sgd escape local minima? In *International conference on machine learning*, pp. 2698–2707. PMLR, 2018.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. Efficient backprop. In *Neural networks: Tricks of the trade*, pp. 9–50. Springer, 2002.

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pp. 8570–8581, 2019.

Lee, J., Schoenholz, S., Pennington, J., Adlam, B., Xiao, L., Novak, R., and Sohl-Dickstein, J. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33:15156–15172, 2020.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Liu, C., Zhu, L., and Belkin, M. On the linearity of large non-linear models: when and why the tangent kernel is constant. *Advances in Neural Information Processing Systems*, 33:15954–15964, 2020.

Liu, C., Zhu, L., and Belkin, M. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 2022a.

Liu, C., Zhu, L., and Belkin, M. Transition to linearity of wide neural networks is an emerging property of assembling weak models. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=CyKHoKyvgnp.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Loo, N., Hasani, R., Amini, A., and Rus, D. Evolution of neural tangent kernels under benign and adversarial training. *Advances in Neural Information Processing Systems*, 35:11642–11657, 2022.

Masters, D. and Luschi, C. Revisiting small batch training for deep neural networks. *arXiv preprint arXiv:1804.07612*, 2018.

Meltzer, D. and Liu, J. Catapult dynamics and phase transitions in quadratic nets. *arXiv preprint arXiv:2301.07737*, 2023.

Myrtle.ai. Myrtle network. https://myrtle.ai/, 2018.

Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, volume 269, pp. 543–547, 1983.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. Exploring generalization in deep learning. *Advances in neural information processing systems*, 30, 2017.

Nguyen, Q., Mukkamala, M. C., and Hein, M. On the loss landscape of a class of deep neural networks with no bad local valleys. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJgXsjA5tQ.

Ortiz-Jiménez, G., Moosavi-Dezfooli, S.-M., and Frossard, P. What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34, 2021.

Papyan, V. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In *International Conference on Machine Learning*, pp. 5012–5021. PMLR, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

Radhakrishnan, A., Beaglehole, D., Pandit, P., and Belkin, M. Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, 383(6690):1461–1467, 2024a.

Radhakrishnan, A., Belkin, M., and Drusvyatskiy, D. Linear recursive feature machines provably recover low-rank matrices. *arXiv preprint arXiv:2401.04553*, 2024b.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pp. 464–472. IEEE, 2017.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pp. 369–386. SPIE, 2019.

Smith, S. L., Dherin, B., Barrett, D., and De, S. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=rq_Qr0c1Hyo.

Trivedi, S., Wang, J., Kpotufe, S., and Shakhnarovich, G. A consistent estimator of the expected gradient outerproduct. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, pp. 819–828, 2014.

Wang, Y., Chen, M., Zhao, T., and Tao, M. Large learning rate tames homogeneity: Convergence and balancing effect. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=3tbDrs77LJ5.

Wang, Z., Li, Z., and Li, J. Analyzing sharpness along gd trajectory: Progressive sharpening and edge of stability. *Advances in Neural Information Processing Systems*, 35: 9983–9994, 2022b.

Wu, L., Zhu, Z., et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. *arXiv preprint arXiv:1706.10239*, 2017.

Xia, Y., Tong, H., Li, W. K., and Zhu, L.-X. An adaptive estimation of dimension reduction space. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(3):363–410, 2002.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Xie, Z., Sato, I., and Sugiyama, M. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=wXgk_iCiYGo.

Xing, C., Arpit, D., Tsirigotis, C., and Bengio, Y. A walk with sgd. *arXiv preprint arXiv:1802.08770*, 2018.

Yang, G. and Hu, E. J. Tensor programs iv: Feature learning in infinite-width neural networks. In *International Conference on Machine Learning*, pp. 11727–11737. PMLR, 2021.

Yuan, G., Xu, M., Kpotufe, S., and Hsu, D. Efficient estimation of the central mean subspace via smoothed gradient outer products. *arXiv preprint arXiv:2312.15469*, 2023.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Zeiler, M. D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, Z. and Xu, Z.-Q. J. Loss spike in training neural networks. *arXiv preprint arXiv:2305.12133*, 2023.

Zhu, L., Liu, C., and Belkin, M. Transition to linearity of general neural networks with directed acyclic graph architecture. *Advances in Neural Information Processing Systems*, 35:5363–5375, 2022.

Zhu, L., Liu, C., Radhakrishnan, A., and Belkin, M. Quadratic models for understanding catapult dynamics of neural networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=PvJnX3dwsD.

Zou, D. and Gu, Q. An improved analysis of training overparameterized deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2053–2062, 2019.

Appendix

A. The critical learning rate can be well approximated using NTK for wide neural networks

In this section, we show that the critical learning rate $\eta_{\text{crit}} := \frac{2}{\lambda_{\max}(H_{\mathcal{L}})}$ can be well approximated using NTK, i.e., $\tilde{\eta}_{\text{crit}} := \frac{n}{\lambda_{\max}(K)(\mathbf{w})}$. Note that $\|K\| = \lambda_{\max}(K)(\mathbf{w})$.

A.1. Approximation of the critical learning rate using NTK during training with a small constant learning rate

For MSE $\mathcal{L}(\mathbf{w}; X) = \frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{w}; \mathbf{x}_i) - y_i)^2$, we can compute its $H_{\mathcal{L}}$ by the chain rule:

$$H_{\mathcal{L}}(\mathbf{w}) = \underbrace{\frac{2}{n}\sum_{i=1}^{n}\left(\frac{\partial f(\mathbf{w}; \mathbf{x}_i)}{\partial \mathbf{w}}\right)^T \frac{\partial f(\mathbf{w}; \mathbf{x}_i)}{\partial \mathbf{w}}}_{\mathcal{A}(\mathbf{w})} + \underbrace{\frac{2}{n}\sum_{i=1}^{n}(f(\mathbf{w}; \mathbf{x}_i) - y_i)\frac{\partial^2 f(\mathbf{w}; \mathbf{x}_i)}{\partial \mathbf{w}^2}}_{\mathcal{B}(\mathbf{w})}.$$

Assume $\|\boldsymbol{x}_i\| = O(1)$ and $|y_i| = O(1)$ for all $i \in [n]$. For $\mathcal{B}(\mathbf{w}_0)$, by random initialization of weights $\mathbf{w}_0$, with high probability, we have $|f(\mathbf{w}_0; \mathbf{x}_i) - y_i| = O(\log m)$, and $\left\|\frac{\partial^2 f(\mathbf{w}_0; \mathbf{x}_i)}{\partial \mathbf{w}^2}\right\|_2 = \tilde{O}(1/\sqrt{m})$ (Liu et al., 2020; Zhu et al., 2022) where m denotes the width of the network. Therefore, by the union bound, with high probability, we have $\mathcal{B}(\mathbf{w}_0) = \tilde{O}(1/\sqrt{m})$.

Note that $\lambda_{\max}(\mathcal{A}(\mathbf{w})) = \frac{2}{n}\lambda_{\max}(K(\mathbf{w}))$ for any $\mathbf{w}$. Combining all the bounds together, we have $\left|\lambda_{\max}(H_{\mathcal{L}})(\mathbf{w}_0) - \frac{2}{n}\lambda_{\max}(K)(\mathbf{w}_0)\right| = \tilde{O}(1/\sqrt{m})$. Then we have

$$|\eta_{\text{crit}} - \tilde{\eta}_{\text{crit}}| = \left|\frac{2}{\lambda_{\max}(H_{\mathcal{L}})(\mathbf{w}_0)} - \frac{n}{\lambda_{\max}(K)(\mathbf{w}_0)}\right| = \tilde{O}(1/\sqrt{m})$$

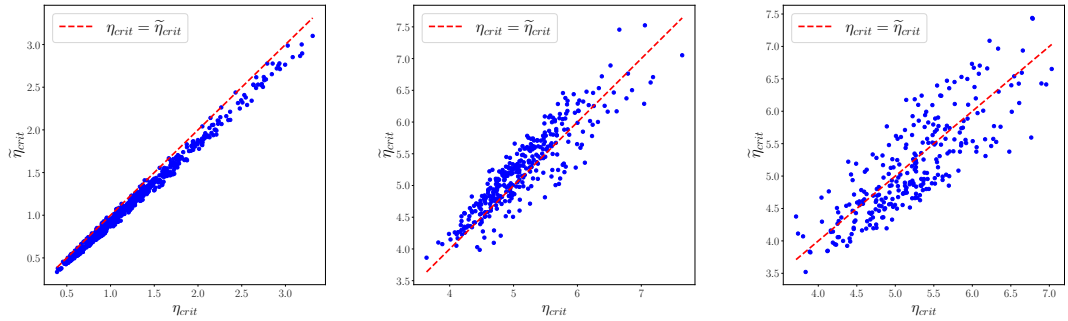
as long as $\lambda_{\max}(K)(\mathbf{w}_0) = \Omega(1)$, which is true with high probability over random initialization for wide networks (Nguyen et al., 2019; Banerjee et al., 2023).

For wide neural networks trained with a small constant learning rate, $\left\|\frac{\partial^2 f(\mathbf{w}_0; \mathbf{x}_i)}{\partial \mathbf{w}^2}\right\|_2 = \tilde{O}(1/\sqrt{m})$ holds during the whole training process of GD/SGD, hence this approximation holds (Liu et al., 2020).

A.2. Approximation of the critical learning rate using NTK during training with a large learning rate

In this section, we provide further evidence for SGD that $\tilde{\eta}_{\text{crit}}$ approximates η_{crit} during training even with a large learning rate. Recall that $\tilde{\eta}_{\text{crit}} = b/\lambda_{\max}(K(\mathbf{w}; X_{\text{batch}}))$ where b is the batch size. We consider the same network architectures as the shallow network in Fig. 6 and deep networks in Fig. 7.

We can see Fig. 12 shows that η_{crit} is close to $\tilde{\eta}_{\text{crit}}$ during training with SGD.



(a) Shallow network (b) 5-layer FCN (c) 5-layer CNN

Figure 12. **Validation of $\eta_{\text{crit}} \approx \tilde{\eta}_{\text{crit}}$ during SGD with catapults.** Plot of points $(\eta_{\text{crit}}, \tilde{\eta}_{\text{crit}})$ at each iteration of SGD for the shallow network, 5-layer FCN and CNN. The models are trained on 128 data points from CIFAR-10 by SGD with batch size 32. The settings are the same with Table 2.

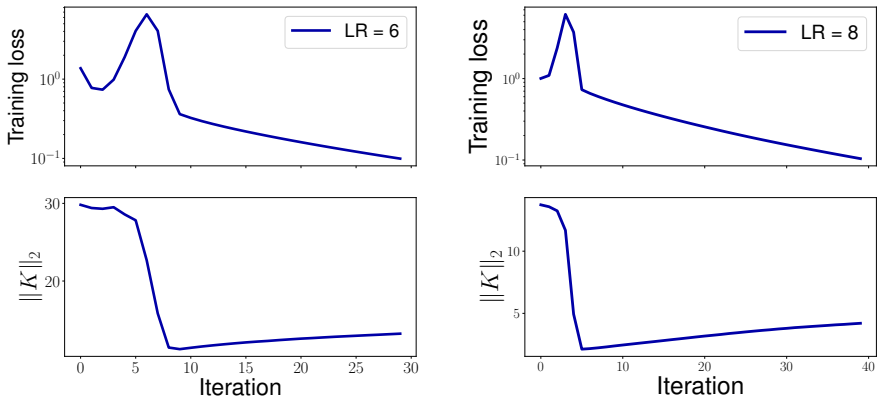
B. Additional experiments for the catapult in GD

B.1. Catapults occur in the top eigenspace of NTK

In this section, we provide additional empirical evidence to verify Claim 1. In particular, we consider three neural network architectures: a 5-layer Fully Connected Neural Network (FCN), a 5-layer Convolutional Neural Network (CNN), and Wide ResNets 10-10; and three datasets CIFAR-10, SVHN, and a synthetic dataset. For the synthetic dataset, we consider the rank-2 regression task with training size 128.

From the experimental results, we can see that for a large learning rate that causes catapult dynamics, the loss spike occurs in the top eigenspace of the tangent kernel. See Fig. 13 for 5-layer FCN and CNN on CIFAR-10 dataset and 14 on SVHN dataset, and 15 for Wide-ResNets on CIFAR-10 dataset.

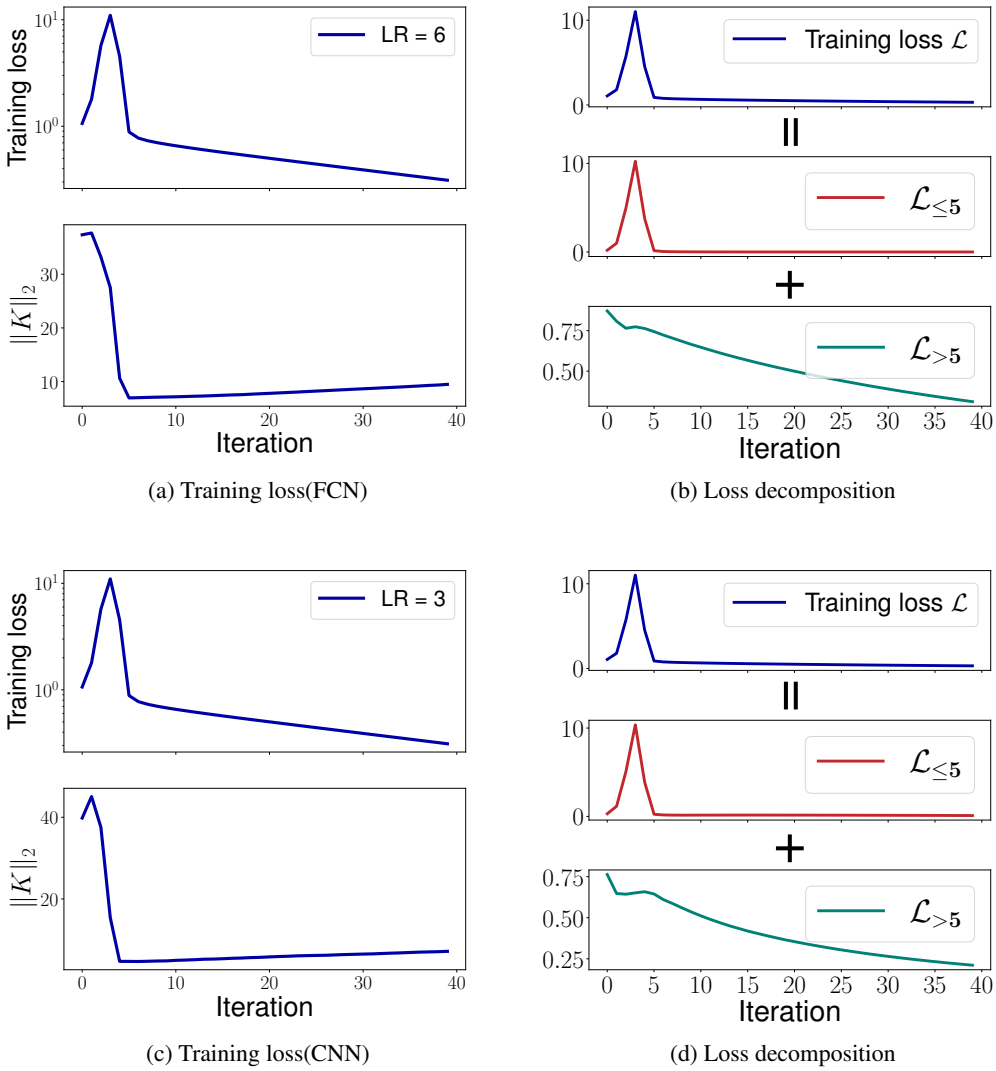
We further show Claim 1 holds for multidimensional outputs in Fig. 16. In particular, for k-class classification tasks, we project the flattened vector of predictions of size kn to the top eigenspaces of the empirical NTK, which is of size $kn \times kn$. Correspondingly, we empirically observe that catapults occur in the top ks eigenspace with a small s.



(a) 5-layer FCN

(b) 5-layer CNN

Figure 13. **The training loss and the spectral norm of the tangent kernel during catapult for 5-layer FCN (a) and CNN (b) on CIFAR-10 dataset**. Both networks are trained under the same experimental setting with Fig. 3.



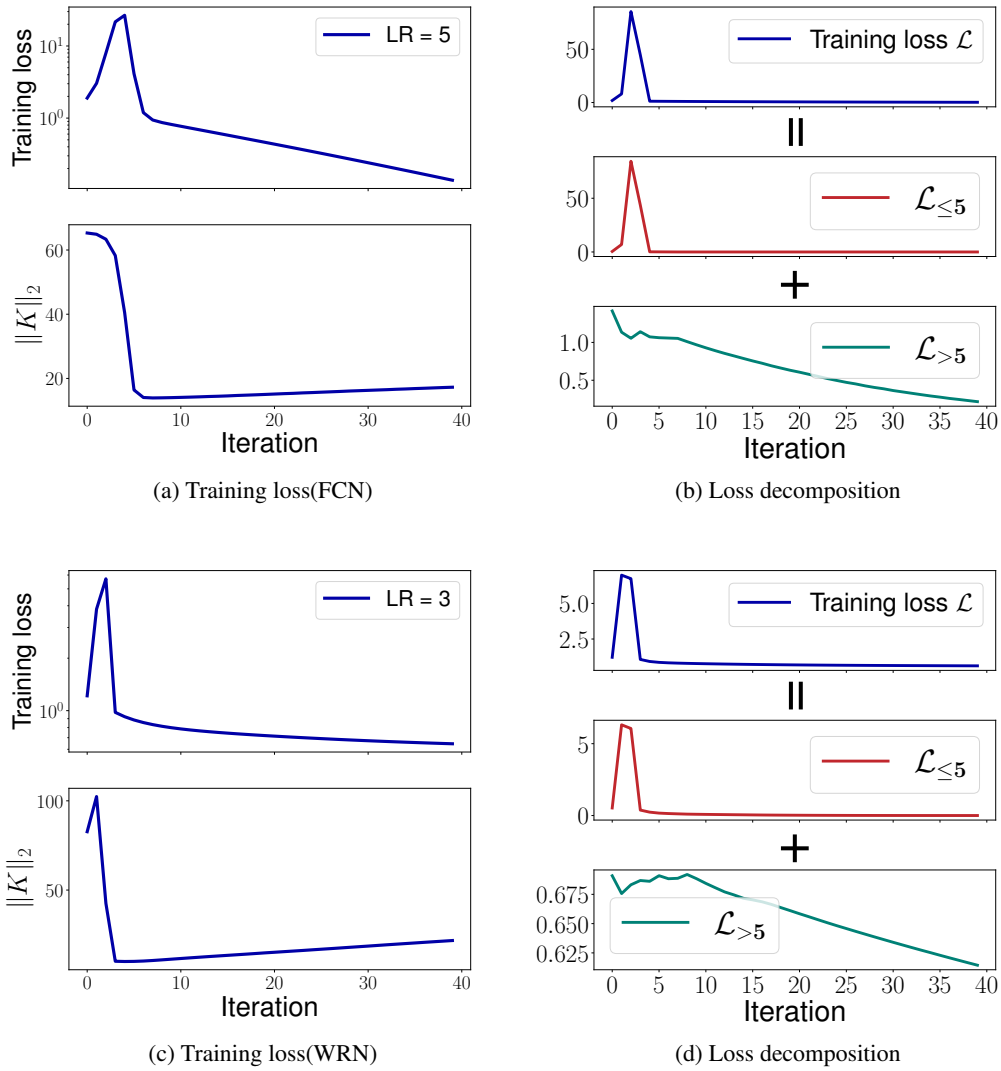
(a) Training loss(FCN)

(b) Loss decomposition

(c) Training loss(CNN)

(d) Loss decomposition

Figure 14. **Catapult dynamics for 5-layer FCN (a-b) and CNN (c-d) on SVHN dataset**. Panel (a) and (c) are the training loss and the spectral norm of the tangent kernel with learning rate 6.0 and 3.0 respectively, and Panel (b) and (d) are the training loss decomposed into the top eigendirections of the tangent kernel, $\mathcal{L}_{\leq 5}$ and the remaining eigendirections, $\mathcal{L}_{>5}$. All the networks are trained on a subset of SVHN with 128 data points. In this experiment, the critical learning rates for FCN and CNN are 3.4 and 1.6 respectively.



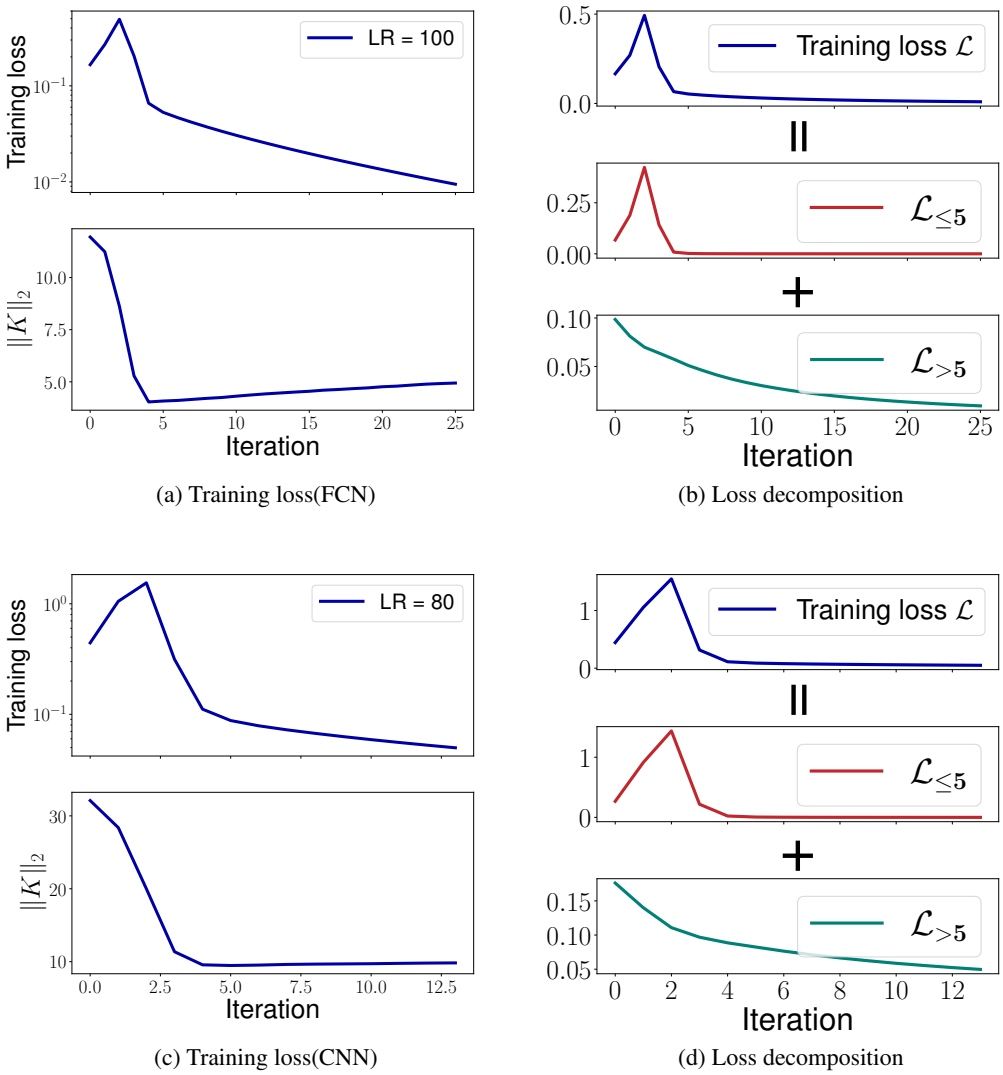
(a) Training loss(FCN)

(b) Loss decomposition

(c) Training loss(WRN)

(d) Loss decomposition

Figure 15. **Catapult dynamics for FCN (a-b) on a synthetic dataset and Wide ResNets 10-10 (c-d) on CIFAR-10 dataset**. Panel (a) and (c) are the training loss and the spectral norm of the tangent kernel with learning rates 5.0 and 3.0 respectively, and Panel (b) and (d) are the training loss decomposed into the top eigendirections of the tangent kernel, $\mathcal{L}_{\leq 5}$ and the remaining eigendirections, $\mathcal{L}_{>5}$. For the synthetic dataset, we use the rank-2 regression task considered in Section 4. The size of the training set is 128. In this experiment, the critical learning rates for FCN and WRN are 1.9 and 1.5 respectively.



(a) Training loss(FCN)

(b) Loss decomposition

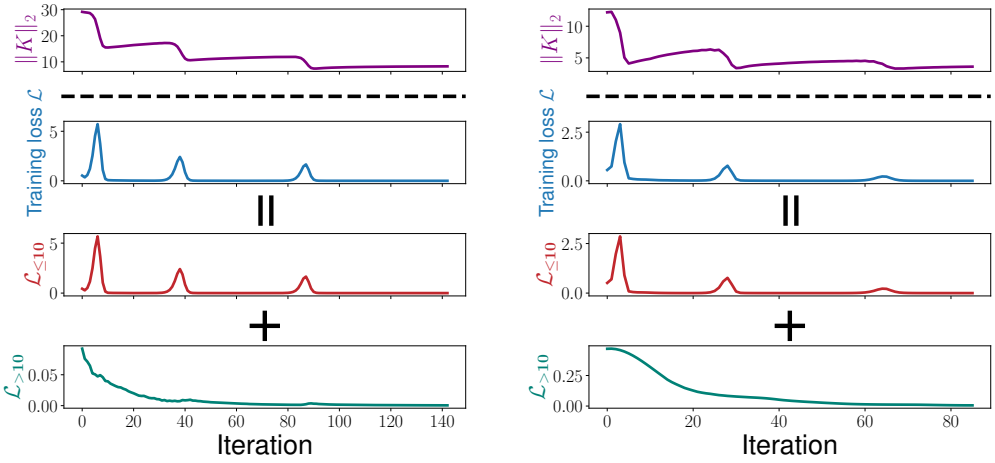
(c) Training loss(CNN)

(d) Loss decomposition

Figure 16. **Catapult dynamics for 5-layer FCN (a-b) and CNN (c-d) on multiclass classification tasks.** Panel (a) and (c) are the training loss and the spectral norm of the tangent kernel with learning rate 100 and 80 respectively, and Panel (b) and (d) are the training loss decomposed into the top eigendirections of the tangent kernel, $\mathcal{L}_{\leq 5}$ and the remaining eigendirections, $\mathcal{L}_{>5}$. All the networks are trained on a subset of CIFAR-10 with 10 classes. Here the dimension of the eigenspace $s = 1, 3, 5$ refers to $10, 30, 50$ respectively due to the output dimension 10. The critical learning rate for FCN and CNN are 34 and 16 respectively.

B.2. Multiple catapults in GD occur in the top eigenspace of NTK

For the multiple catapults shown in Fig. 4, similar to a single catapult, we show that the catapults occur in the top eigenspace of NTK. See Fig. 17.



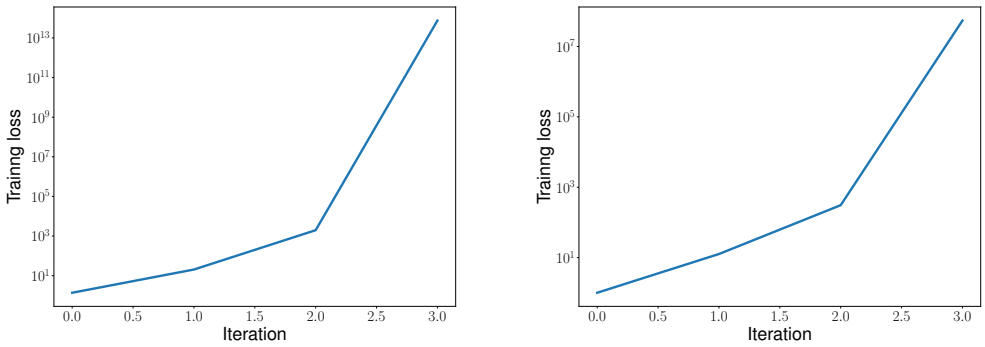
(a) 5-layer FCN

(b) 5-layer CNN

Figure 17. **Multiple catapults in GD with increased learning rates.** With the same setting of Fig. 4, the training loss is decomposed into the top eigendirections of the tangent kernel, $\mathcal{L}_{\leq 10}$ and the remaining eigendirections, $\mathcal{L}_{>10}$.

B.3. Multiple catapults allow a larger learning rate at convergence

Corresponding to Fig. 4 in Section 3.2, we show that if the neural networks are trained with the learning rate at the convergence, i.e., after multiple catapults, the GD will diverge.



(a) 5-layer FCN

(b) 5-layer CNN

Figure 18. **GD diverges when trained with the learning rate after multiple catapults.** Corresponding to Fig. 4, we train the model using GD with learning rate at convergence, 60 and 40 respectively for the 5-layer FCN and CNN.

C. Additional experiments for catapults in SGD

C.1. Loss decomposition for shallow network corresponding to Fig. 6

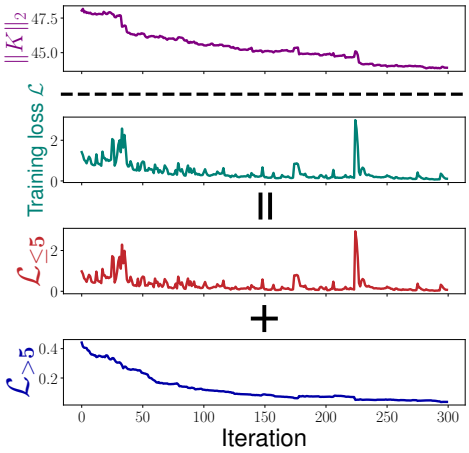
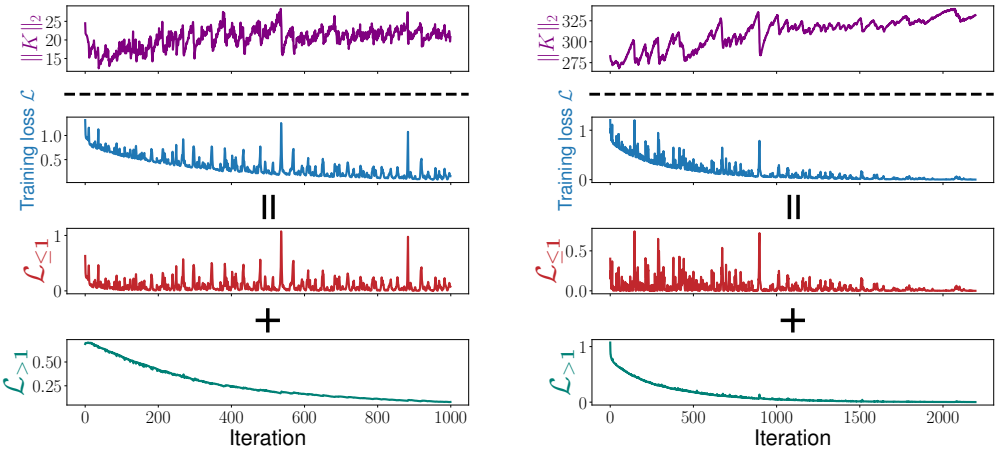


Figure 19. **The spectral norm of NTK and the loss decomposition during catapults for shallow network.** This figure corresponds to Fig. 6 with learning rate 0.8. The training loss is decomposed into the component corresponding to the top-5 eigendirections of the NTK and the other components.

C.2. Full training process visualization corresponding to Fig. 7

We present the complete training loss and the spectrum norm of the NTK corresponding to Fig. 7(c,d) in Fig. 20.



(a) Wide ResNets 10-10

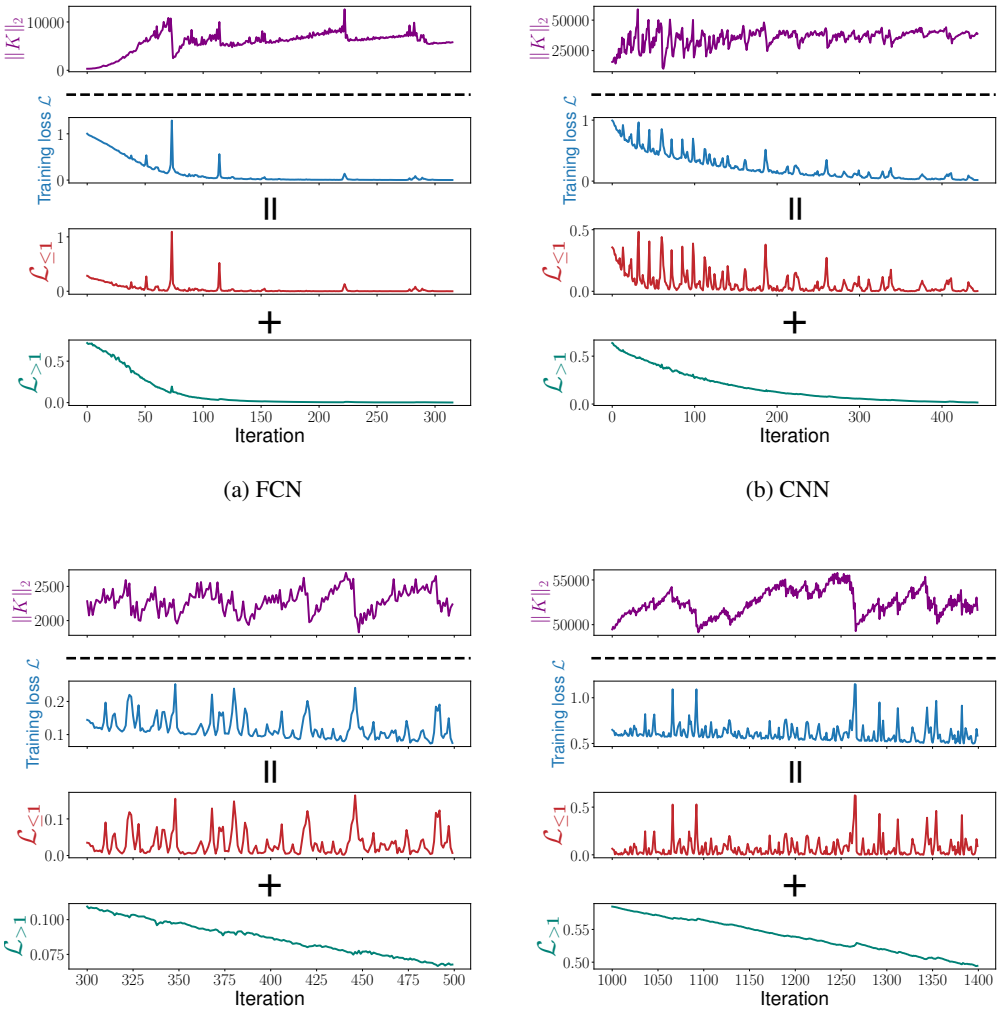
(b) ViT-4

Figure 20. **Cataput dynamics in SGD for modern deep architectures.** The complete versions corresponding to Fig. 7(c,d). The training loss is decomposed based on the eigendirections of the NTK: $\mathcal{L}_{\leq 1}$ and $\mathcal{L}_{>1}$.

C.3. Catapults in SGD with Pytorch default parameterization

In Fig. 7, we used NTK parameterization (see the definition in Appendix F) for the neural networks. We further validate our empirical observations on (1) the occurrence of the loss spikes of SGD in the top eigenspace of the tangent kernel and (2) the

decrease in the spectral norm of the tangent kernel during loss spikes in the setting with Pytorch default parameterization, under which the wide networks are still close to their linear approximations (Liu et al., 2020; Yang & Hu, 2021) in Fig. 21.



(a) FCN

(b) CNN

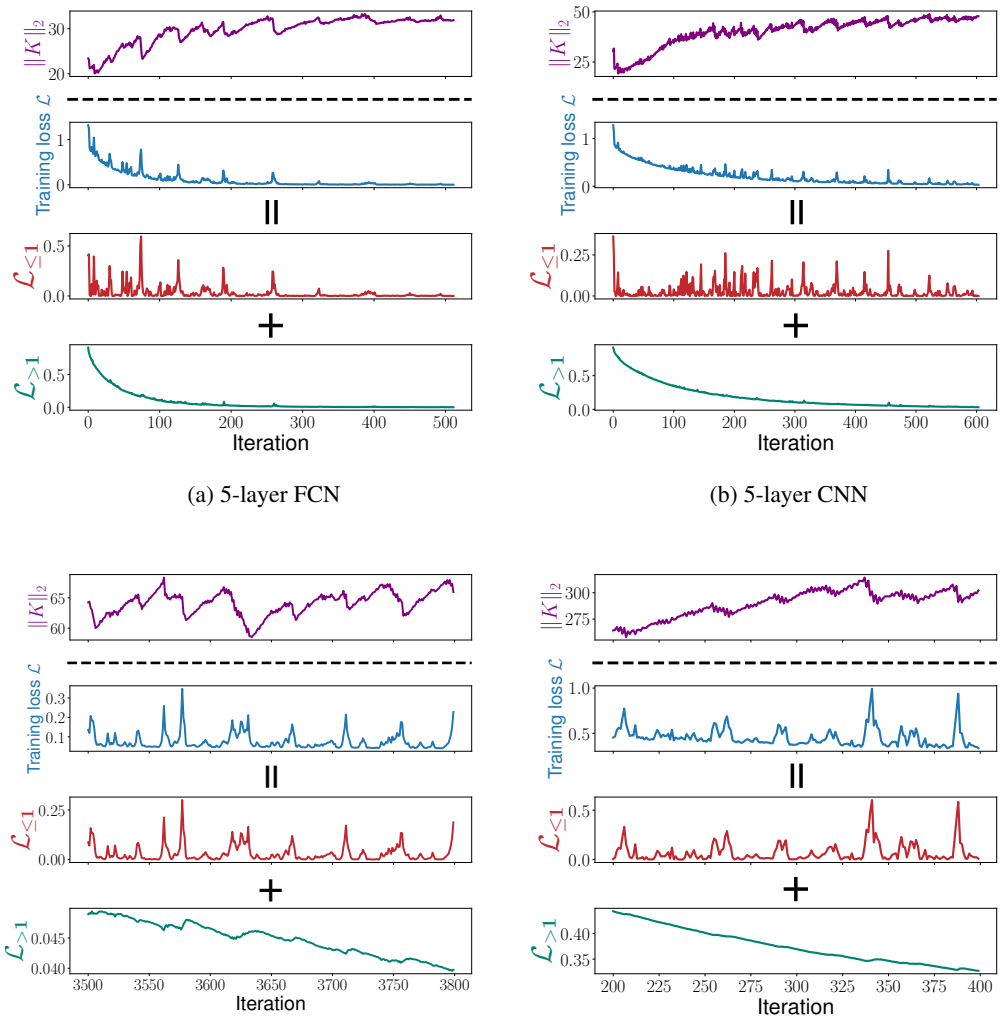
(c) Wide ResNets 10-10

(d) ViT-4

Figure 21. **Cataput dynamics in SGD for modern deep architectures with Pytorch default parameterization.** The tasks are the same with Fig. 7 except that we use Pytorch default parameterization. The training loss is decomposed based on the eigendirections of the NTK: $\mathcal{L}_{\leq 1}$ and $\mathcal{L}_{>1}$.

C.4. Catapults in SGD with additional datasets

We show that the findings in Fig. 7 hold for a subset of SVHN dataset (see Fig. 22) and for a larger dataset ($5,000$ data points from CIFAR-2) and for multi-class classification problems (see Fig. 23).



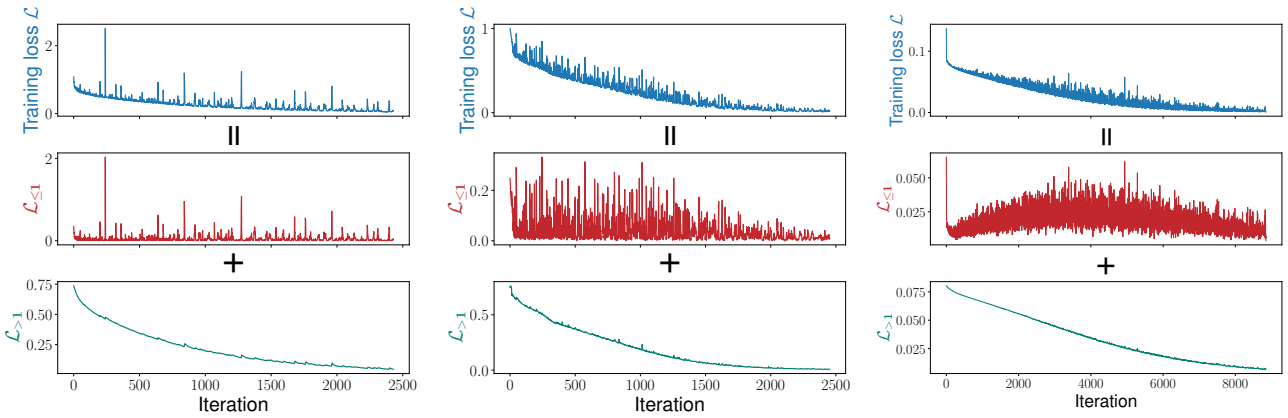
(a) 5-layer FCN

(b) 5-layer CNN

(c) Wide ResNets 10-10 (zoomed-in)

(d) ViT-4 (zoomed-in)

Figure 22. **Cataput dynamics in SGD for modern deep architectures on 2-class SVHN.** The tasks are the same with Fig. 7 except that we train the neural networks on a subset of SVHN dataset. The training loss is decomposed into the top eigenspace of the tangent kernel $\mathcal{L}_{\leq 1}$ and its complement $\mathcal{L}_{>1}$. Here $\mathcal{L} = \mathcal{L}_{\leq 1} + \mathcal{L}_{>1}$.

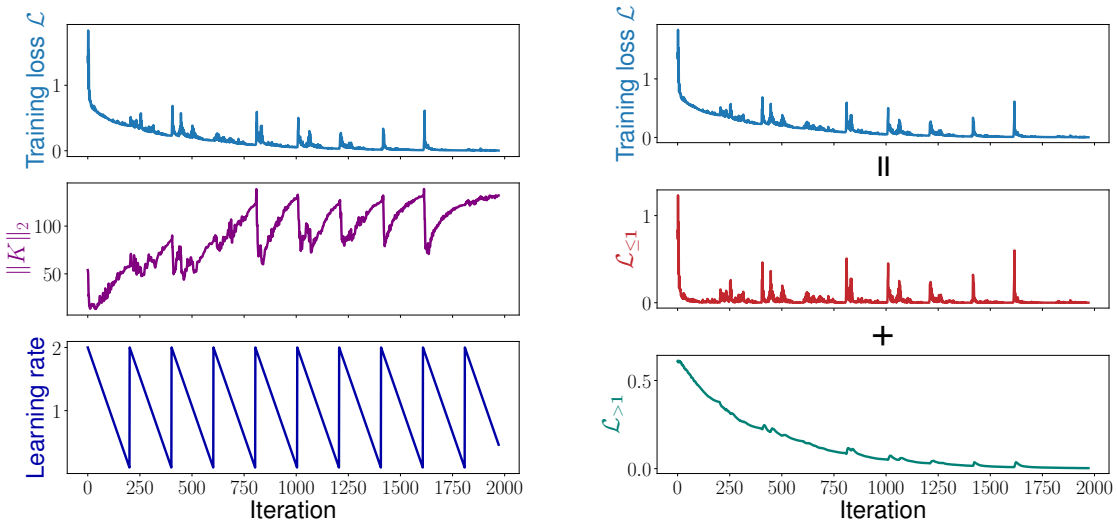


(a) 5-layer FCN (b) 5-layer CNN (c) Myrtle network (Myrtle.ai, 2018)

Figure 23. **Catapult dynamics in SGD for large datasets (Panel (a) and (b)) and multi-class classification problems (Panel(c)).** Panel(a,b): The networks are trained on $5,000$ data points from CIFAR-2. Panel(c): The network is trained on 128 points from CIFAR-10. The training loss is decomposed into the top eigenspace of the tangent kernel $\mathcal{L}_{\leq 1}$ and its complement $\mathcal{L}_{>1}$. Here $\mathcal{L} = \mathcal{L}_{\leq 1} + \mathcal{L}_{>1}$.

C.5. Catapults occur in training with cyclical learning rates

In this section, we show that catapults occur in SGD with a cyclical learning rate schedule. Specifically, we show that loss spikes occur in the top eigenspace of the tangent kernel and there is a decrease in the spectral norm of the tangent kernel according to each loss spike.



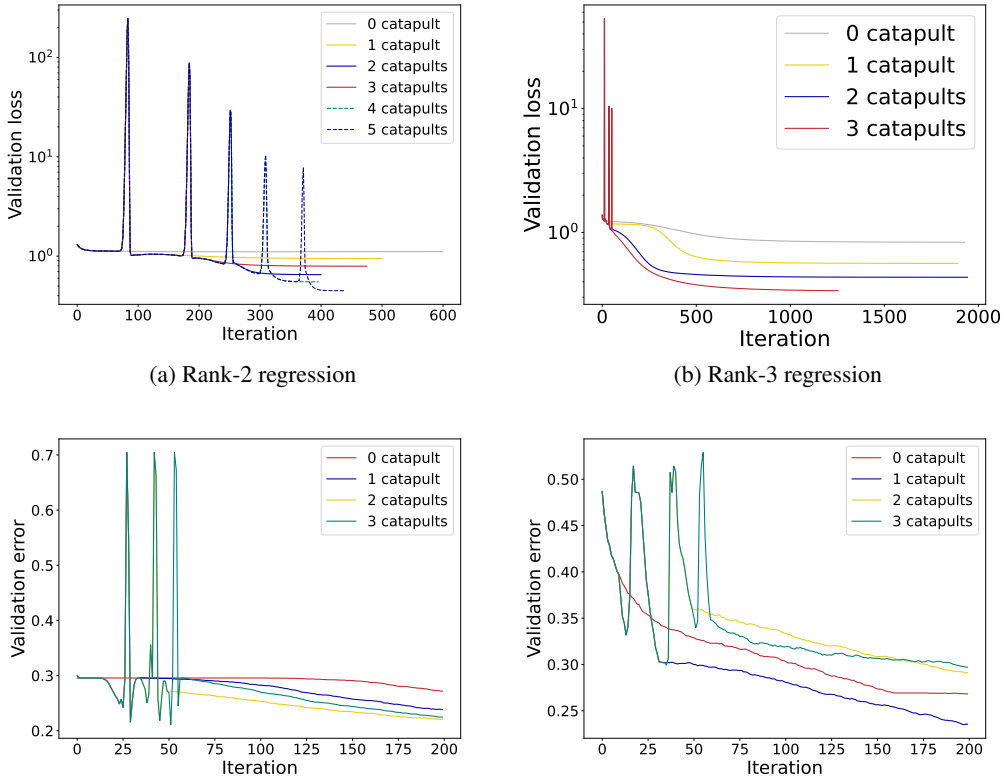
(a) Training loss, $\|K\|_2$ and learning rate (b) Loss decomposition

Figure 24. **Catapults in SGD with cyclical learning rates**. Panel (a): The plot of the training loss and the spectral norm of the tangent kernel corresponding to the whole training set with a cyclic learning rate schedule. Panel (b): The training loss is decomposed into the top and non-top eigenspace of the tangent kernel, i.e.,$\mathcal{L}_{\leq 1}$ and $\mathcal{L}_{>1}$. Here $\mathcal{L} = \mathcal{L}_{\leq 1} + \mathcal{L}_{>1}$. We train Wide ResNets 10-10 on a subset of CIFAR-10. The setting is the same with Fig. 7c except for a cyclical learning rate schedule.

D. Additional experiments for feature learning in GD

D.1. Validation loss/error for multiple catapults corresponding to Fig. 8

We present the validation loss/error in Fig. 25 for the tasks corresponding to Fig. 8. The learning rate is increased during training to generate multiple catapults.



(a) Rank-2 regression

(b) Rank-3 regression

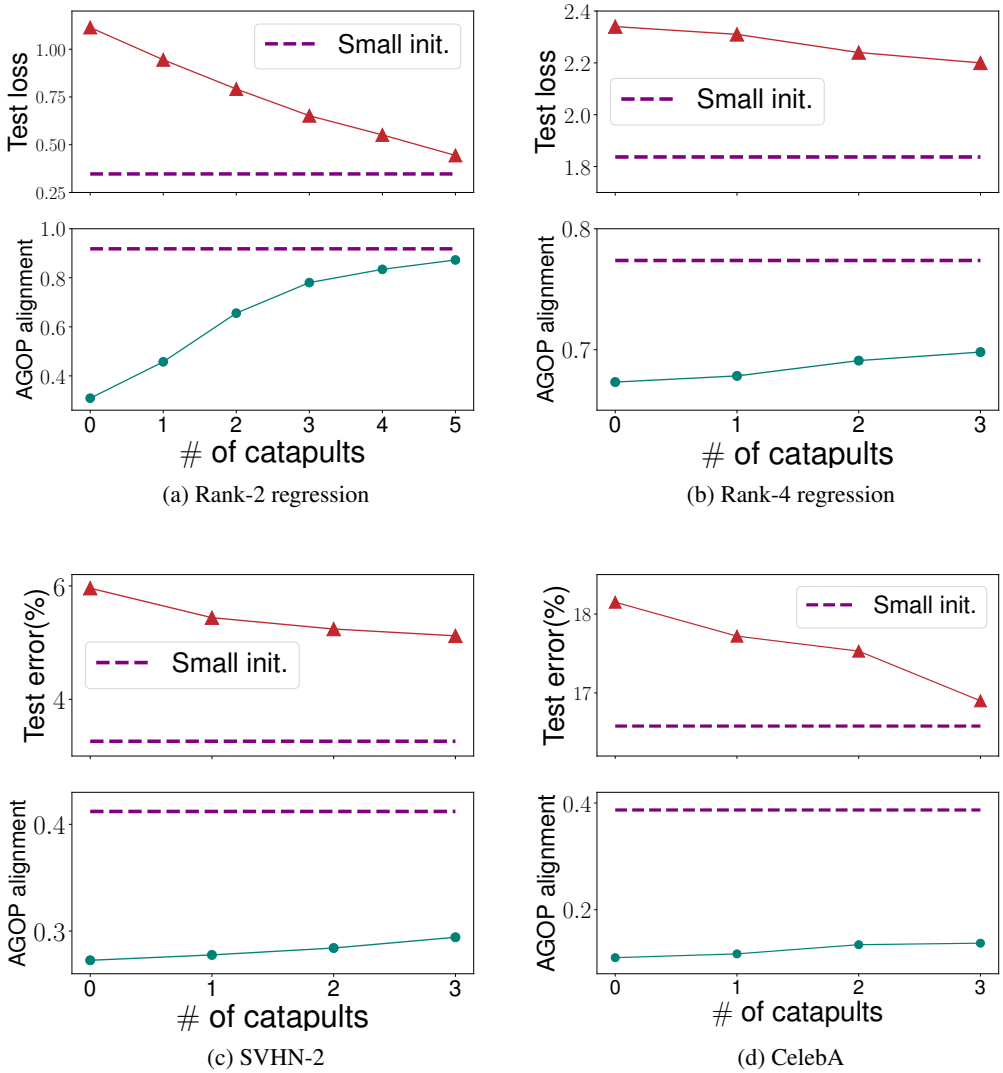
(c) SVHN-2

(d) CelebA

Figure 25. **Validation loss/error of multiple catapults in GD corresponding to** Fig. 8. Panel(c)&(d) only present first 200 iterations.

D.2. Feature learning with near zero initialization

We compare the performance of networks exhibiting multiple catapults with those initialized using near zero initialization scheme, i.e., each weight is sampled i.i.d. from $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.1$. This is in contrast to the NTK parameterization where we use $\sigma = 1$. It was argued in (Yang & Hu, 2021) that feature learning occurs with near zero initialization. We can see that small initialization achieves the smallest test loss/error as well as the best AGOP alignment, which indicates that learning AGOP correlates strongly with the test performance.



(a) Rank-2 regression

(b) Rank-4 regression

(c) SVHN-2

(d) CelebA

Figure 26. **Multiple catapults in GD compared to the small initialization scheme.** We train a 2-layer FCN in Panel(a), a 4-layer FCN in Panel(b,d) and a 5-layer CNN in Panel(c). For small initialization, each weight parameter is i.i.d. from $\mathcal{N}(0, \sigma^2)$ with $\sigma = 0.1$. The experimental setup is the same as Fig. 8.

For the Rank-2 regression task, we visualize the AGOP in the following Fig. 27, where we can see that the features are learned better, i.e., closer to the True AGOP, with a greater number of catapults.

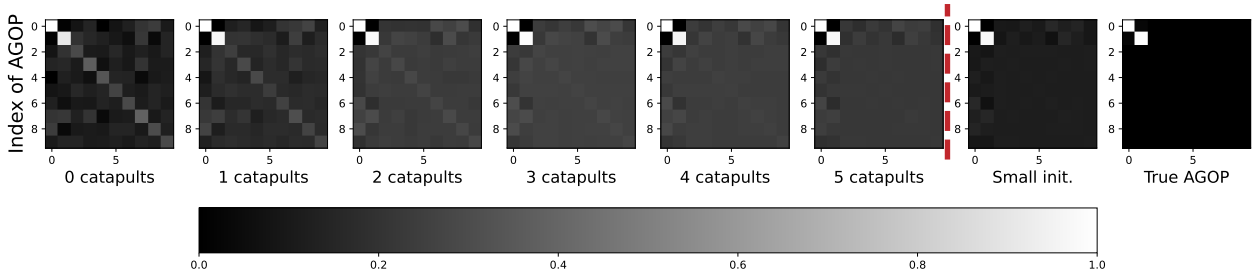
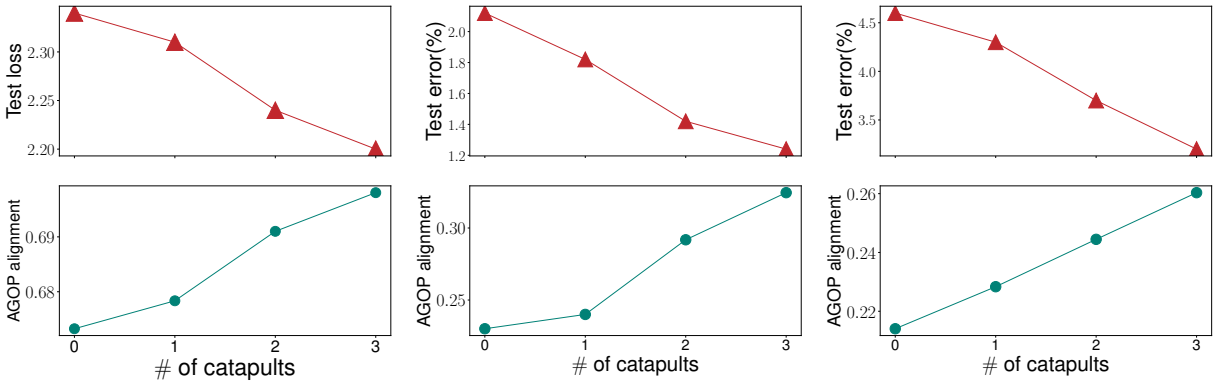


Figure 27. **Visualization of AGOP for rank-2 regression task.** All pixels are normalized to the range $[0, 1]$ and the top 10 rows and columns of the AGOP are plotted.

D.3. Feature learning in GD for additional datasets

In this section, we show the findings observed in Fig. 8 hold for Rank-4 regression, USPS dataset and Fashion MNIST dataset. See Fig. 28.

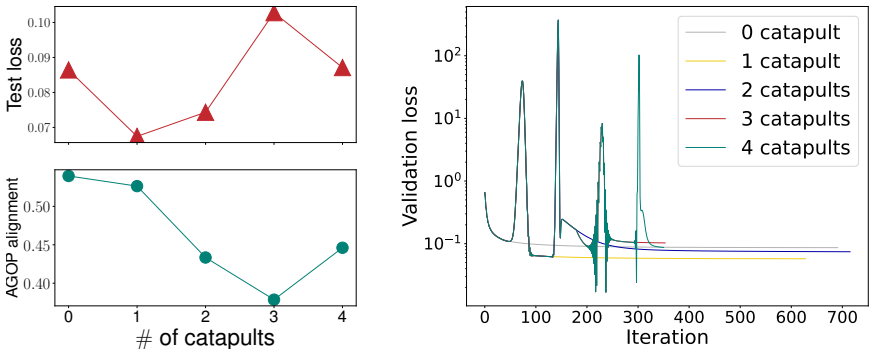


(a) Rank-4 regression (b) USPS (c) Fashion MNIST

Figure 28. **Correlation between AGOP alignment and test performance in GD with multiple catapults on additional datasets.** We train a 4-layer FCN using GD for all tasks. The learning rate is increased multiple times during training to generate multiple catapults. Experimental details can be found in Appendix F.4.

D.4. No feature learning for full rank task

In Fig. 29, we show that for a full-rank task where the target function is $f^*(\boldsymbol{x}) = \frac{1}{\sqrt{d}}\|\boldsymbol{x}\|$, catapults do not improve the test performance or the AGOP alignment.



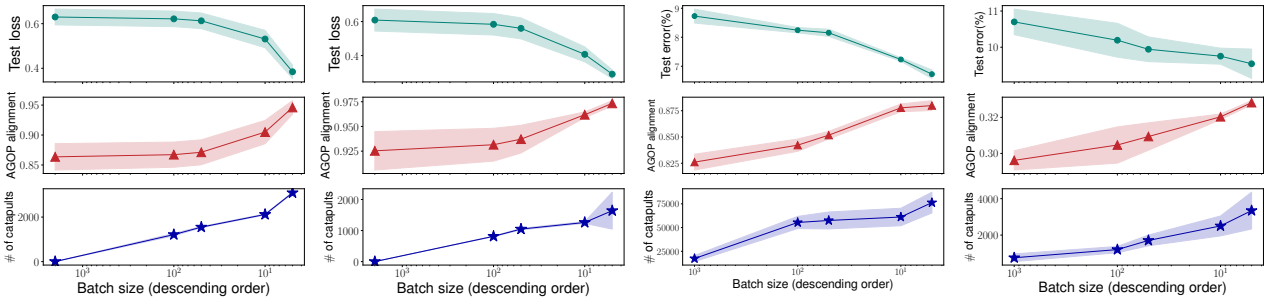
(a) Validation loss vs. iteration (b) Test loss and AGOP alignment

Figure 29. **Multiple catapults in GD for a full rank task.** We train a 2-layer FCN on a synthetic dataset with a full rank target function $f^*(\boldsymbol{x}) = \frac{1}{\sqrt{d}}\|\boldsymbol{x}\|$ using GD.The learning rate is increased multiple times during training to generate multiple catapults. The experimental details can be found in Appendix F.6.

E. Additional experiments for feature learning in SGD

E.1. Feature learning of catapults in SGD with Pytorch parameterization

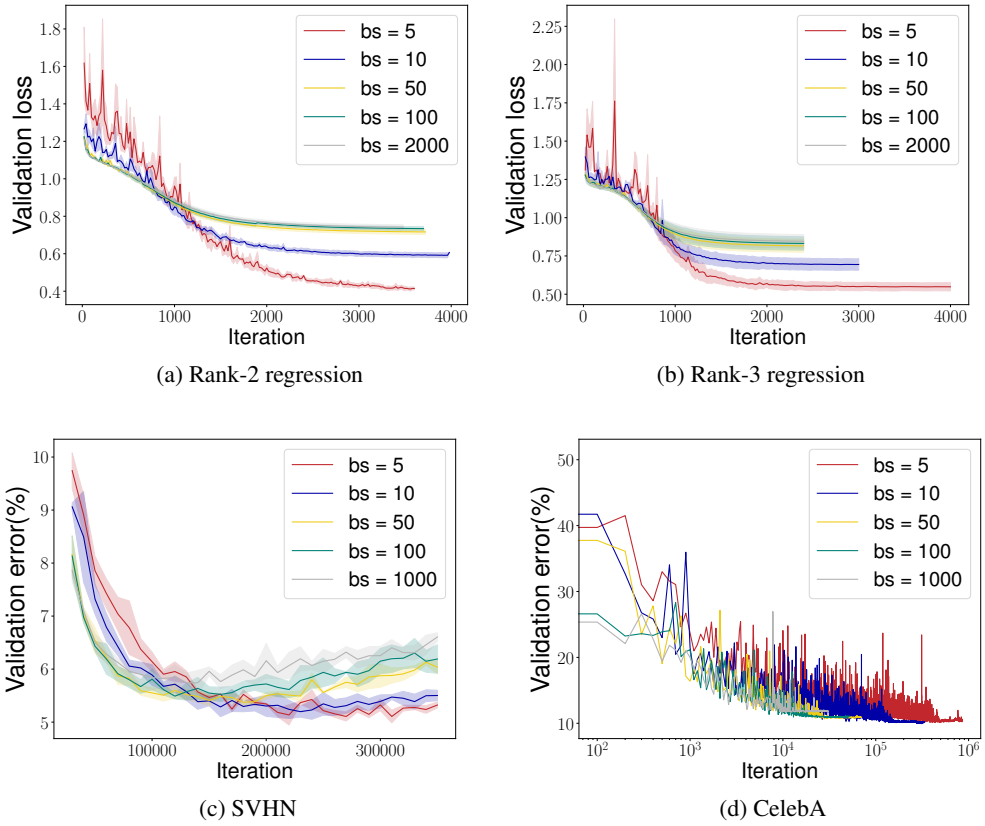
In this section, we further verify our observation on the feature learning of SGD with Pytorch default parameterization on the same tasks with Fig. 9 in Section 4.



(a) Rank-2 regression (b) Rank-3 regression (c) SVHN (d) CelebA

Figure 30. **Correlation between AGOP alignment and test performance in SGD with Pytorch default parameterization.** The tasks are the same with Fig. 9 except that we use Pytorch default parameterization.

E.2. Validation loss/error of SGD corresponding to Fig. 9 and 30



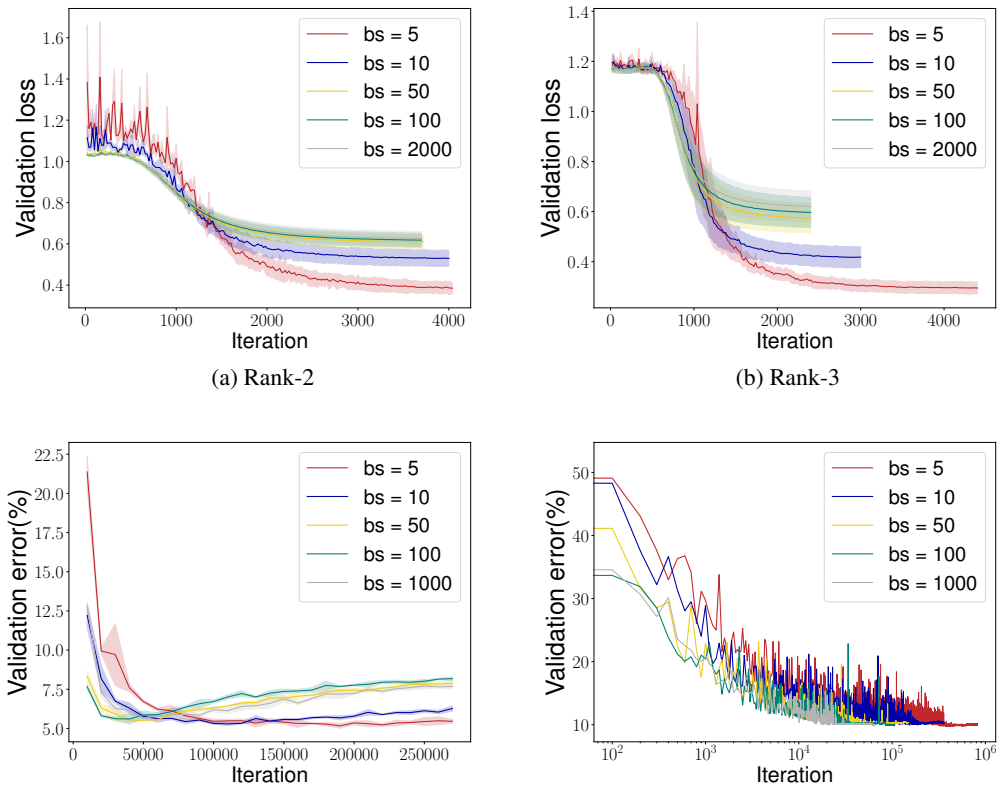
(a) Rank-2 regression

(b) Rank-3 regression

(c) SVHN

(d) CelebA

Figure 31. **Validation loss/error corresponding to** Fig. 9. Panel(c) presents the validation error from iteration 4000.



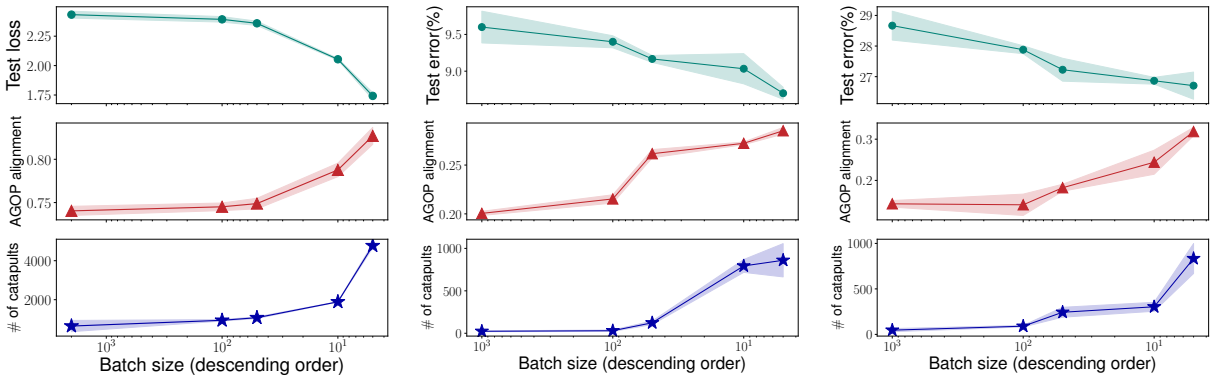
(a) Rank-2 (b) Rank-3

(c) SVHN (d) CelebA

Figure 32. **Validation loss/error with Pytorch default parameterization corresponding to** Fig. 30. Panel(c) presents the validation error from iteration 2000.

E.3. Feature learning in SGD for additional datasets

In this section, we show the findings observed in Fig. 9 hold for Rank-4 regression, USPS dataset and Fashion MNIST dataset. See Fig. 33.

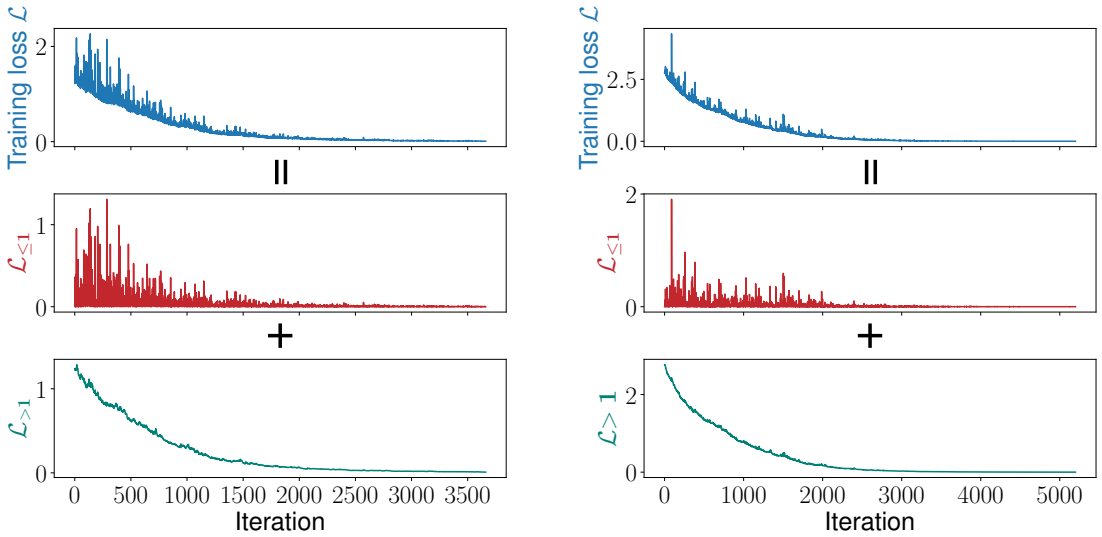


(a) Rank-4 regression (b) USPS (c) Fashion MNIST

Figure 33. **Correlation between AGOP alignment and test performance in SGD.** We train a 4-layer fully connected neural network using SGD.

E.4. Verification of catapults in SGD

In Fig. 34, we verify that the spikes in the training loss of SGD with small batch sizes are caused by catapult dynamics. Specifically, we show that the spikes occur in the top eigendirection of the NTK.



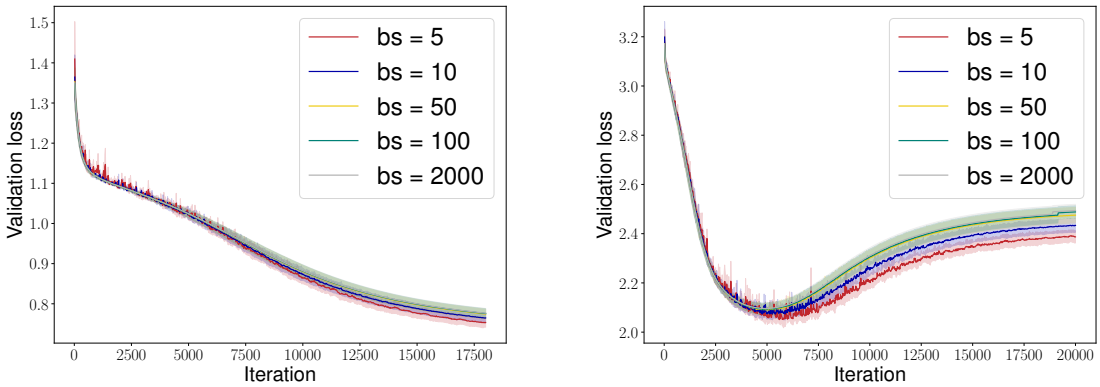
(a) Loss decomposition (Rank-2)

(b) Loss decomposition (Rank-4)

Figure 34. **Verification of catapult dynamics: loss decomposition of Rank-2 and Rank-4 regression tasks corresponding to Fig. 9 with batch size** 5**.** The training loss is decomposed into the top eigenspace of the tangent kernel $\mathcal{L}_{\leq 1}$ and its complement $\mathcal{L}_{>1}$. Here $\mathcal{L} = \mathcal{L}_{\leq 1} + \mathcal{L}_{>1}$.

E.5. No feature learning with a small learning rate for SGD

In Fig. 9, we have shown that a smaller batch size leads to more catapults, hence resulting in better test performance. In this section, we show that the test performance with different batch sizes is similar when training with a small learning rate, where no catapults occur. This further verifies that a greater number of catapults accounts for better test performance for small batch sizes. See Fig. 35.



(a) Rank-2 (b) Rank-4

Figure 35. **The networks are trained with a smaller learning rate corresponding to Fig. 31a &b.** We consider the same setting as in Fig. 31 except for selecting a smaller learning rate $\eta_{\text{crit}}/40$ compared to $\eta_{\text{crit}}/2$ in Fig. 31. Here η_{crit} is the critical learning rate for the whole dataset.

F. Experimental details

For all the networks considered in this paper, we use ReLU activation functions. We parameterize the networks by NTK parameterization (Jacot et al., 2018). Note that NTK parametrization is widely used for understanding neural networks (Lee et al., 2020; Du et al., 2019; Lewkowycz et al., 2020). We also verify our results with Pytorch (Paszke et al., 2019) default parameterization for the experiments shown in Fig. 21 and 30.

NTK parameterization. Given a neural network with NTK parameterization, all the trainable weight parameters are i.i.d. from $\mathcal{N}(0,1)$. For a fully connected layer, it takes the form $f^{\ell+1} = \text{ReLU}\left(\frac{1}{\sqrt{m_\ell}} W^\ell f^\ell + b^\ell\right)$ where $W^\ell \in \mathbb{R}^{m_{\ell+1}\times m_\ell}, f^\ell \in \mathbb{R}^{m_\ell}, b^\ell \in \mathbb{R}^{m_{\ell+1}}$. For a convolutional layer, it takes the form $f^{\ell+1}_{i,j,k} = \text{ReLU}\left(\frac{1}{\sqrt{m_\ell s^2}} \sum_{p=0}^{\lceil \frac{s+1}{2} \rceil} \sum_{q=0}^{\lceil \frac{s+1}{2} \rceil} \sum_{o=1}^{m_\ell} W^\ell_{p,q,o,k} f^\ell_{i-\lceil \frac{s-1}{2} \rceil, j-\lceil \frac{s-1}{2} \rceil, o} + b^\ell_k\right)$, where $W^\ell \in \mathbb{R}^{s\times s\times m_\ell \times m_{\ell+1}}, f^\ell \in \mathbb{R}^{h\times w\times m_\ell}, b^\ell \in \mathbb{R}^{m_{\ell+1}}$. Note that s is the filter size and we assume the stride to be 1 in this case. For f^ℓ with negative indices, we let it be 0, i.e., zero padding. For the output layer, we use a linear layer without activation functions.

Dataset. For the synthetic datasets, we generate data $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ by i.i.d. $\boldsymbol{x}_i \sim \mathcal{N}(0, I_{100})$ and $y_i = f^*(\boldsymbol{x}) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.1^2)$. For two real-world datasets, we consider a subset of CelebA dataset with glasses as the label, the Street View House Numbers (SVHN) dataset, USPS dataset and Fashion MNIST dataset. Due to computational limitations with GD, for some tasks, we select two classes (number 0 and 2) of SVHN dataset, USPS dataset and Fashion MNIST dataset.

EGOP (Epexcted Gradient Outer Product). Note that for these low-rank polynomial regression tasks, we know the analytical form of target functions hence we can calculate the EGOP by $G^* = \mathbb{E}_{\boldsymbol{x}} \frac{\partial f^*}{\partial \boldsymbol{x}} \frac{\partial f^*}{\partial \boldsymbol{x}}^T$. For real-world datasets, we estimate the EGOP by using the AGOP of one of the state-of-the-art models $\hat{f}$ that achieve high test accuracy: $\hat{G} = \frac{1}{n} \sum_{i=1}^n \frac{\partial \hat{f}}{\partial \boldsymbol{x}_i} \frac{\partial \hat{f}}{\partial \boldsymbol{x}_i}^T$.

In the following, we provide the detailed experimental setup for each experiment. Note that in the classification tasks, i.e. CelebA and SVHN datasets, the test error refers to the classification error on the test split.

F.1. Experiments in Section 3.1

Fig. 3: We use a 2-class subset of CIFAR-10 dataset (Krizhevsky et al., 2009) (class 7 and class 9) and randomly select 128 data points out of it. For the network architectures, we use a 5-layer FCN with width 1024 and 5-layer CNN with 512 channels per layer. For CNN, we flatten the image into a one-dimensional vector before the last fully connected layer.

F.2. Experiments in Section 3.2

Fig. 4: We use the same training tasks as in Fig. 3. For FCN, we start with a learning rate 6 and we increase the learning rate to $[10, 15]$ at iteration $[15, 60]$. For CNN, we start with a learning rate 8 and we increase the learning rate to $[15, 20]$ at iteration $[10, 40]$.

F.3. Experiments in Section 3.3

Fig. 5: We consider a synthetic dataset (x_i, y_i) where x_i are sampled i.i.d. on unit sphere and $y_i = 1$ with training size and dimension both equal to 100. We train a wide two-layer ReLU network with second-layer weights fixed, using SGD with batch size one. The critical learning rate $\eta_{\rm crit}(x_i)$ for (a minibatch of size one) x_i is proportional to $1/\|x_i\|_2^2 = const$. We select one data point (x_*, y_*) from the training set and multiply both x_* and y_* by 2. This makes the critical learning rate corresponding to the sample x_* four times smaller. We choose the (constant) learning rate η between the critical learning rate of (the mini-batch) x_* and the critical learning rates of the rest of the data points. Thus SGD with learning rate η induces catapult on x_* but not on any other data points.

Fig. 6: For the shallow network, we use a 2-layer FCN with width 1024. We train the model on 128 data points from CIFAR 2 using SGD with batch size 32. We use a constant learning rate 0.8. We stop training when the training loss is less than 10^{-3}.

Table 2: The 5-layer FCN and CNN are the same as in Fig. 3. We train the model on 128 data points from CIFAR 2 using SGD with batch size 32. We use a constant learning rate 6 and 8 for 5-layer FCN and CNN respectively. We stop training when the training loss is less than 10^{-3}.

Fig. 7: The 5-layer FCN and CNN are the same as in Fig. 3. And we use the standard Wide ResNets 10-10 and ViT-4 architectures. The learning rates for 5-layer FCN, 5-layer CNN, are $6, 8, 3, 0.2$ respectively. We train the model with a constant learning rate, and we stop training when the training loss is less than 10^{-3}. All the models are trained on 128 data points from CIFAR-2 using SGD with batch size 32.

F.4. Experiments in Section 4

Fig. 8: For rank-2 task, we train a 2-layer FCN with width 1024. The size of the training set, testing set and validation set are $2000, 5000$ and 5000 respectively.

For rank-3 task, CelebA tasks, we train a 4-layer FCN with width 256. The size of the training set, testing set and validation set are $1000, 5000$ and 5000 respectively.

For SVHN-2 tasks, we train a 5-layer CNN with width 256. We select class 0 and class 2 out of the full SVHN datasets as SVHN-2. The size of the training set, testing set and validation set are $1000, 5000$ and 5000 respectively.

We increase the learning rate during training. For Rank-2 task, we increase the learning rate to $[8, 16, 30, 50, 75, 80]$ at iteration $[50, 150, 220, 280, 350, 400]$. For Rank-3 task, we increase the learning rate to $[40, 100, 150]$ at iteration $[20, 60, 80]$. For SVHN-2 task, we increase the learning rate to $[30, 60, 90]$ at iteration $[10, 35, 50]$. For CelebA task, we increase the learning rate to $[40, 70, 100]$ at iteration $[10, 35, 50]$. We decay the learning rate if necessary after the catapult to avoid extra catapults until the end of training.

Fig. 9: For both Rank-2 and Rank-3 tasks, we let the size of training set, testing set and validation set be $2000, 5000$ and 5000. For the SVHN task, we train the full SVHN using the 5-layer Myrtle network. For the CelebA task, we train the full 2-class CelebA dataset with glasses feature using 4-layer FCN with width 256. To obtain the true AGOP , we use one of the SOTA models (WideResNet 16-2) which achieves 97.2% test accuracy on SVHN and 5-layer Myrtle network which achieves 95.7% test accuracy on CelebA.

We use the same learning rate across batch sizes for each task. The learning rate is chosen as $\frac{1}{2}\eta_{\rm crit}$ corresponding to the whole training set. For SVHN and CelebA tasks, we estimate $\eta_{\rm crit}$ using a subset with size 5000 of the whole training set. We train the model with batch size $[5, 10, 50, 100, 2000]$. For all tasks, we stop training when the training loss is less than 10^{-3}. We report the average of 3 independent runs.

Fig. 11: We use the same network architectures and training/validation/testing sets as in Fig. 8.

For all the tasks, except for GD, all the optimizers use a mini-batch size 100.

We stop training when the training loss is less than 10^{-3}. We report the average of 3 independent runs.

For the rank-2 task and rank-4 task, we know the target function hence we can analytically compute the exact true AGOP . For SVHN-2 task and CelebA task, to estimate the true AGOP , we use one of the SOTA models, Myrtle-5 which achieves 98.4% test accuracy on two-class SVHN dataset and 95.7% test accuracy on CelebA dataset.

The following table is the learning rate we choose for the experiments:

Task	SGD	GD	SGD+M	Adadelta	Adagrad	RMSprop	Adam
Rank-2	2.0	2.0	2.0	2.0	0.1	10^{-2}	10^{-2}
Rank-3	2.0	2.0	2.0	2.0	10^{-2}	10^{-2}	10^{-3}
Rank-4	1.0	1.0	1.0	1.0	5×10^{-3}	10^{-3}	10^{-3}
SVHN-2	5.0	5.0	5.0	5.0	5×10^{-3}	10^{-4}	10^{-3}
CelebA	10.0	10.0	10.0	10.0	5×10^{-3}	10^{-3}	10^{-3}

Table 3. **Choice of learning rates for Fig. 11.**

The experiment is to demonstrate the correlation between AGOP alignment and test performance. For this reason, we did not fine-tune the learning rate to achieve the best test performance.

F.5. Experiments in Appendix C

Fig. 16: We use the same network architectures as in Fig. 3 and we train 128 data point from CIFAR-10.

Fig. 22: We use the same setting as Fig. 7, except we train the networks on 128 data points from SVHN-2(number 0 and 2).

Fig. 23: For panel(a) and panel(b), we train the same 5-layer FCN and CNN as in Fig. 3 and on 5,000 data points from CIFAR-2. For panel(c), we train a 5-layer Myrtle network on 128 points from CIFAR-10.

F.6. Experiments in Appendix D

Fig. 28: For rank-4 task, USPS dataset and Fashion MNIST dataset, we train a 4-layer FCN with width 256. The size of the training set, testing set and validation set are $1000, 5000$ and 5000 respectively.

For rank-4 task, we increase the learning rate to $[15, 40, 60]$ at iteration $[50, 75, 110]$. For USPS dataset, we increase the learning rate to $[15, 30, 40]$ at iteration $[10, 30, 45]$. For Fashion MNIST dataset, we increase the learning rate to $[10, 40, 55]$ at iteration $[6, 20, 30]$.

Fig. 29: We train a 2-layer FCN with width 1024. We consider a synthetic dataset, where $f^*(\boldsymbol{x}) = \frac{1}{\sqrt{d}}\|\boldsymbol{x}\|$. The size of the training set and validation set is $128, 2000$ respectively. During training, we start with the learning rate 6 and increase the learning rate to $[7, 12, 40, 80]$ at iteration $[30, 120, 180, 280]$.

F.7. Experiments in Appendix E

Fig. 30: We use the same setup with Fig. 9 except that all the networks are parameterized with Pytorch default parameterization. The learning rates are 0.01, 0.01, 0.05 and 1.0 for each task.

Fig. 33: For rank-4 task, USPS dataset and Fashion MNIST dataset, we train a 4-layer FCN with width 256. The size of the training set, testing set and validation set are $2000, 5000$ and 5000 respectively. We add 10% label noise for the USPS

dataset and Fashion MNIST dataset. To obtain the true AGOP , we use one of the SOTA models (5-layer CNN) which achieves 99.2% test accuracy on USPS and 5-layer Myrtle network which achieves 91.8% test accuracy on Fashion MNIST.